libNeuroML Documentation

libNeuroML authors and contributors

Oct 17, 2022

CONTENTS

Welcome to the libNeuroML documentation. Here you will find information on installing, using, and contributing to libNeuroML. For more information on NeuroML standard, other tools in the NeuroML eco-system, the NeuroML community and how to get in touch with us, please see the documentation at https://docs.neuroml.org.

CHAPTER

ONE

USER GUIDE

1.1 Introduction

This package provides Python libNeuroML, for working with neuronal models specified in NeuroML 2.

Warning: libNeuroML targets NeuroML v2.0

libNeuroML targets NeuroML v2.0, which is described in Cannon et al, 2014). NeuroML v1.8.1 (Gleeson et al. 2010) is now deprecated and not supported by libNeuroML.

For a detailed description of libNeuroML see Vella *et al.* [VCC+14]. *Please cite the paper if you use libNeuroML.*

1.1.1 NeuroML

NeuroML provides an object model for describing neuronal morphologies, ion channels, synapses and 3D network structure. For more information on NeuroML 2 and LEMS please see the NeuroML documentation.

1.1.2 Serialisations

The XML serialisation will be the "natural" serialisation and will follow closely the NeuroML object model. The format of the XML will be specified by the XML Schema definition (XSD file).

Other serialisations have also been developed (HDF5, SWC). Please see Vella *et al.* [VCC+14] for more details.

1.2 Installation

1.2.1 Using Pip

On most systems with a Python installation, libNeuroML can be installed using the default Python package manager, Pip:

```
pip install libNeuroML
```

It is recommended to use a virtual environment when installing Python packages using *pip* to prevent these from conflicting with other system libraries.

This will support the default XML serialization. To install all of requirements to include the other serialisations, use

```
# On Ubuntu based systems
sudo apt-get install libhdf5-dev
pip install libNeuroML[full]
```

The `apt` line is required at time of writing because PyTables' wheels for python 3.7 depend on the system libhdf5.

1.2.2 On Fedora based systems

On Fedora Linux systems, the NeuroFedora community provides libNeuroML in the standard Fedora repos and can be installed using the following commands:

```
sudo dnf install python3-libNeuroML
```

1.2.3 Install from source

You can clone the GitHub repository and also build libNeuroML from the sources. For this, you will need git:

```
git clone git://github.com/NeuralEnsemble/libNeuroML.git
cd libNeuroML
```

More details about the git repository and making your own branch/fork are here. To build and install libNeuroML, you can use the standard install method for Python packages (preferably in a virtual environment):

```
python setup.py install
```

To use the **latest development version of libNeuroML**, switch to the development branch:

```
git checkout development
sudo python setup.py install
```

1.2.4 Run an example

Some sample scripts are included in *neuroml/examples*, e.g. :

```
cd neuroml/examples
python build_network.py
```

The standard examples can also be found *Examples*.

1.2.5 Unit tests

To run unit tests cd to the directory *neuroml/test* and use the Python unittest module discover method:

```
cd neuroml/test/
python -m unittest discover
```

If all tests passed correctly, your output should look something like this:

```
.......
----------------------------------------------------------------------
Ran 55 tests in 40.1s

OK
```

You can also use PyTest to run tests.

```
pip install pytest
pytest -v --strict -W all
```

1.3 API documentation

The libNeuroML API includes the core NeuroML classes and various utilities. You can find information on these in the pages below.

1.3.1 `nml` Module (NeuroML Core classes)

These NeuroML core classes are Python representations of the Component Types defined in the NeuroML standard . These can be used to build NeuroML models in Python, and these models can then be exported to the standard XML NeuroML representation. These core classes also contain some utility functions to make it easier for users to carry out common tasks.

Each NeuroML Component Type is represented here as a Python class. Due to implementation limitations, whereas NeuroML Component Types use lower camel case naming, the Python classes here use upper camel case naming. So, for example, the `adExIaFCell` Component Type in the NeuroML schema becomes the `AdExIaFCell` class here, and `expTwoSynapse` becomes the `ExpTwoSynapse` class.

The `child` and `children` elements that NeuroML Component Types can have are represented in the Python classes as variables. The variable names, to distinguish them from class names, use snake case. So for example, the `cell` NeuroML Component Type has a corresponding `Cell` Python class here. The `biophysicalProperties` child Component Type in `cell` is represented as the `biophysical_properties` list variable in the `Cell` Python class. The class signatures list all the child/children elements and text fields that the corresponding Component Type possesses. To again use the `Cell` class as an example, the construction signature is this:

```
class neuroml.nml.nml.Cell(neuro_lex_id=None, id=None, metaid=None, notes=None,␣
↪properties=None, annotation=None, morphology_attr=None, biophysical_properties_
↪attr=None, morphology=None, biophysical_properties=None, extensiontype_=None, **kwargs_
↪)
```

As can be seen here, it includes both the `biophysical_properties` and `morphology` child elements as variables.

Please see the examples in the NeuroML documentation to see usage examples of libNeuroML. Please also note that this module is also included in the top level of the *neuroml* package, so you can use these classes by importing neuroml:

```
from neuroml import AdExIaFCell
```

List of Component classes

This documentation is auto-generated from the NeuroML schema. In case of issues, please refer to the schema documentation for clarifications. If the schema documentation does not resolve the issue, please contact us.

GeneratedsSuperSuper

class neuroml.nml.generatedssupersuper.**GeneratedsSuperSuper**

Bases: object

Super class for GeneratedsSuper.

Any bits that must go into every libNeuroML class should go here.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GdsCollector

class neuroml.nml.generatedscollector.**GdsCollector**(*messages=None*)

Bases: object

add_message(*msg*)

clear_messages()

get_messages()

print_messages()

write_messages(*outstream*)

AdExIaFCell

class neuroml.nml.nml.**AdExIaFCell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *C: a Nml2Quantity_capacitance (required) = None*, *g_l: a Nml2Quantity_conductance (required) = None*, *EL: a Nml2Quantity_voltage (required) = None*, *reset: a Nml2Quantity_voltage (required) = None*, *VT: a Nml2Quantity_voltage (required) = None*, *thresh: a Nml2Quantity_voltage (required) = None*, *del_t: a Nml2Quantity_voltage (required) = None*, *tauw: a Nml2Quantity_time (required) = None*, *refract: a Nml2Quantity_time (required) = None*, *a: a Nml2Quantity_conductance (required) = None*, *b: a Nml2Quantity_current (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseCellMembPotCap*

AdExIaFCell – Model based on Brette R and Gerstner W (2005) Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity. J Neurophysiol 94:3637-3642

Parameters

- **gL** (*conductance*) –
- **EL** (*voltage*) –
- **VT** (*voltage*) –
- **thresh** (*voltage*) –
- **reset** (*voltage*) –
- **delT** (*voltage*) –
- **tauw** (*time*) –
- **refract** (*time*) –
- **a** (*conductance*) –
- **b** (*current*) –
- **C** (*capacitance*) – Total capacitance of the cell membrane

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child

elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

AlphaCondSynapse

class neuroml.nml.nml.**AlphaCondSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *tau_syn: a float (required) = None*, *e_rev: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BasePynnSynapse*

AlphaCondSynapse – Alpha synapse: rise time and decay time are both tau_syn. Conductance based synapse.

Parameters

- **e_rev** (*none*) –
- **tau_syn** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

AlphaCurrSynapse

class neuroml.nml.nml.**AlphaCurrSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *tau_syn: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BasePynnSynapse*

AlphaCurrSynapse – Alpha synapse: rise time and decay time are both tau_syn. Current based synapse.

Parameters
tau_syn (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

AlphaCurrentSynapse

class neuroml.nml.nml.**AlphaCurrentSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *tau: a Nml2Quantity_time (required) = None*, *ibase: a Nml2Quantity_current (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseCurrentBasedSynapse*

AlphaCurrentSynapse – Alpha current synapse: rise time and decay time are both **tau.**

Parameters

- **tau** (*time*) – Time course for rise and decay
- **ibase** (*current*) – Baseline current increase after receiving a spike

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided,

only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
`return_format` (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
`recursive` (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
`ValueError` – if component is invalid

AlphaSynapse

class neuroml.nml.nml.**AlphaSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *gbase: a Nml2Quantity_conductance (required) = None*, *erev: a Nml2Quantity_voltage (required) = None*, *tau: a Nml2Quantity_time (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: BaseConductanceBasedSynapse

AlphaSynapse – Ohmic synapse model where rise time and decay time are both **tau.** Max conductance reached during this time (assuming zero conductance before) is **gbase** * **weight.**

Parameters

- **tau** (*time*) – Time course of rise/decay
- **gbase** (*conductance*) – Baseline conductance, generally the maximum conductance following a single spike
- **erev** (*voltage*) – Reversal potential of the synapse

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Annotation

class neuroml.nml.nml.**Annotation**(*anytypeobjs_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Annotation – A structured annotation containing metadata, specifically RDF or **property** elements

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Base

class neuroml.nml.nml.**Base**(*id: a NmlId (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Base – Anything which can have a unique (within its parent) id of the form NmlId (spaceless combination of letters, numbers and underscore).

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child

elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BaseCell

class neuroml.nml.nml.**BaseCell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

BaseCell – Base type of any cell (e. g. point neuron like **izhikevich2007Cell** , or a morphologically detailed **Cell** with **segment** s) which can be used in a **population**

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found

- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BaseCellMembPotCap

class neuroml.nml.nml.**BaseCellMembPotCap**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *C: a Nml2Quantity_capacitance (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseCell*

BaseCellMembPotCap – Any cell with a membrane potential **v** with voltage units and a membrane capacitance **C.** Also defines exposed value **iSyn** for current due to external synapses and **iMemb** for total transmembrane current (usually channel currents plus **iSyn**)

Parameters
C (*capacitance*) – Total capacitance of the cell membrane

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BaseConductanceBasedSynapse

class neuroml.nml.nml.**BaseConductanceBasedSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *gbase: a Nml2Quantity_conductance (required) = None*, *erev: a Nml2Quantity_voltage (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseVoltageDepSynapse*

BaseConductanceBasedSynapse – Synapse model which exposes a conductance **g** in addition to producing a current. Not necessarily ohmic!! cno_0000027

Parameters

- **gbase** (*conductance*) – Baseline conductance, generally the maximum conductance following a single spike
- **erev** (*voltage*) – Reversal potential of the synapse

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BaseConductanceBasedSynapseTwo

class neuroml.nml.nml.**BaseConductanceBasedSynapseTwo**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *gbase1: a Nml2Quantity_conductance (required) = None*, *gbase2: a Nml2Quantity_conductance (required) = None*, *erev: a Nml2Quantity_voltage (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseVoltageDepSynapse*

BaseConductanceBasedSynapseTwo – Synapse model suited for a sum of two expTwoSynapses which exposes a conductance **g** in addition to producing a current. Not necessarily ohmic!! cno_0000027

Parameters

- **gbase1** (*conductance*) – Baseline conductance 1
- **gbase2** (*conductance*) – Baseline conductance 2
- **erev** (*voltage*) – Reversal potential of the synapse

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BaseConnection

class neuroml.nml.nml.**BaseConnection**(*id: a NmlId (required) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseNonNegativeIntegerId*

BaseConnection – Base of all synaptic connections (chemical/electrical/analog, etc.) inside projections

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BaseConnectionNewFormat

class neuroml.nml.nml.**BaseConnectionNewFormat**(*id: a NmlId (required) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *pre_cell: a string (required) = None*, *pre_segment: a NonNegativeInteger (optional) = '0'*, *pre_fraction_along: a ZeroToOne (optional) = '0.5'*, *post_cell: a string (required) = None*, *post_segment: a NonNegativeInteger (optional) = '0'*, *post_fraction_along: a ZeroToOne (optional) = '0.5'*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseConnection*

BaseConnectionNewFormat – Base of all synaptic connections with preCell, postSegment, etc. See BaseConnectionOldFormat

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)

- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BaseConnectionOldFormat

class neuroml.nml.nml.**BaseConnectionOldFormat**(*id: a NmlId (required) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *pre_cell_id: a string (required) = None*, *pre_segment_id: a NonNegativeInteger (optional) = '0'*, *pre_fraction_along: a ZeroToOne (optional) = '0.5'*, *post_cell_id: a string (required) = None*, *post_segment_id: a NonNegativeInteger (optional) = '0'*, *post_fraction_along: a ZeroToOne (optional) = '0.5'*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseConnection*

BaseConnectionOldFormat – Base of all synaptic connections with preCellId, postSegmentId, etc. Note: this is not the best name for these attributes, since Id is superfluous, hence BaseConnectionNewFormat

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BaseCurrentBasedSynapse

class neuroml.nml.nml.**BaseCurrentBasedSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseSynapse*

BaseCurrentBasedSynapse – Synapse model which produces a synaptic current.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child

elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BaseNonNegativeIntegerId

class neuroml.nml.nml.**BaseNonNegativeIntegerId**(*id: a NmlId (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

BaseNonNegativeIntegerId – Anything which can have a unique (within its parent) id, which must be an integer zero or greater.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BaseProjection

class neuroml.nml.nml.**BaseProjection**(*id: a NmlId (required) = None*, *presynaptic_population: a NmlId (required) = None*, *postsynaptic_population: a NmlId (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

BaseProjection – Base for projection (set of synaptic connections) between two populations

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BasePynnSynapse

class neuroml.nml.nml.**BasePynnSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *tau_syn: a float (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseSynapse*

BasePynnSynapse – Base type for all PyNN synapses. Note, the current **I** produced is dimensionless, but it requires a membrane potential **v** with dimension voltage

Parameters
tau_syn (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
`return_format` (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
`recursive` (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
`ValueError` – if component is invalid

BaseSynapse

class neuroml.nml.nml.**BaseSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

BaseSynapse – Base type for all synapses, i. e. ComponentTypes which produce a current (dimension current) and change Dynamics in response to an incoming event. cno_0000009

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BaseVoltageDepSynapse

class neuroml.nml.nml.**BaseVoltageDepSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseSynapse*

BaseVoltageDepSynapse – Base type for synapses with a dependence on membrane potential

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BaseWithoutId

class neuroml.nml.nml.**BaseWithoutId**(*extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *GeneratedsSuper*

BaseWithoutId – Base element without ID specified *yet*, e.g. for an element with a particular requirement on its id which does not comply with NmlId (e.g. Segment needs nonNegativeInteger).

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously

- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *Member-Spec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BiophysicalProperties

class neuroml.nml.nml.**BiophysicalProperties**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *membrane_properties: a MembraneProperties (required) = None*, *intracellular_properties: a IntracellularProperties (optional) = None*, *extracellular_properties: a ExtracellularProperties (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

BiophysicalProperties – The biophysical properties of the **cell** , including the **membraneProperties** and the **intracellularProperties**

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found

- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BiophysicalProperties2CaPools

class neuroml.nml.nml.**BiophysicalProperties2CaPools**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *membrane_properties2_ca_pools: a MembraneProperties2CaPools (required) = None*, *intracellular_properties2_ca_pools: a IntracellularProperties2CaPools (optional) = None*, *extracellular_properties: a ExtracellularProperties (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

BiophysicalProperties2CaPools – The biophysical properties of the **cell** , including the **membraneProperties2CaPools** and the **intracellularProperties2CaPools** for a cell with two Ca pools

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided,

only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BlockMechanism

class neuroml.nml.nml.**BlockMechanism**(*type: a BlockTypes (required) = None*, *species: a NmlId (required) = None*, *block_concentration: a Nml2Quantity_concentration (required) = None*, *scaling_conc: a Nml2Quantity_concentration (required) = None*, *scaling_volt: a Nml2Quantity_voltage (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need

to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

BlockingPlasticSynapse

class neuroml.nml.nml.**BlockingPlasticSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *gbase: a Nml2Quantity_conductance (required) = None*, *erev: a Nml2Quantity_voltage (required) = None*, *tau_decay: a Nml2Quantity_time (required) = None*, *tau_rise: a Nml2Quantity_time (required) = None*, *plasticity_mechanism: a PlasticityMechanism (optional) = None*, *block_mechanism: a BlockMechanism (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *ExpTwoSynapse*

BlockingPlasticSynapse – Biexponential synapse that allows for optional block and plasticity mechanisms, which can be expressed as child elements.

Parameters

- **tauRise** (*time*) –
- **tauDecay** (*time*) –
- **gbase** (*conductance*) – Baseline conductance, generally the maximum conductance following a single spike
- **erev** (*voltage*) – Reversal potential of the synapse

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Case

class neuroml.nml.nml.**Case**(*condition: a string (optional) = None*, *value: a string (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *GeneratedsSuper*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an

information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Cell

`class neuroml.nml.nml.Cell`(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *morphology_attr: a NmlId (optional) = None*, *biophysical_properties_attr: a NmlId (optional) = None*, *morphology: a Morphology (optional) = None*, *biophysical_properties: a BiophysicalProperties (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseCell*

Cell – Cell with **segment** s specified in a **morphology** element along with details on its **biophysicalProperties** . NOTE: this can only be correctly simulated using jLEMS when there is a single segment in the cell, and **v** of this cell represents the membrane potential in that isopotential segment.

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

add_channel_density(*nml_cell_doc*, *cd_id*, *ion_channel*, *cond_density*, *erev='0.0 mV'*, *group_id='all'*, *ion='non_specific'*, *ion_chan_def_file=''*)

Add channel density.

Parameters

- **nml_cell_doc** (NeuroMLDocument) – cell NeuroML document to which channel density is to be added
- **cd_id** (*str*) – id for channel density
- **ion_channel** (*str*) – name of ion channel
- **cond_density** (*str*) – value of conductance density with units
- **erev** (*str*) – value of reversal potential with units
- **group_id** (*str*) – segment groups to add to
- **ion** (*str*) – name of ion
- **ion_chan_def_file** (*str*) – path to NeuroML2 file defining the ion channel, if empty, it assumes the channel is defined in the same file

add_channel_density_v(*channel_density_type*, *nml_cell_doc*, *ion_chan_def_file=''*, ***kwargs*)

Generic function to add channel density components to a Cell.

Parameters

- **channel_density_type** (*str*) – type of channel density to add. See https://docs.neuroml.org/Userdocs/Schemas/Cells.html for the complete list.
- **nml_cell_doc** (NeuroMLDocument) – cell NeuroML document to which channel density is to be added
- **ion_chan_def_file** (*str*) – path to NeuroML2 file defining the ion channel, if empty, it assumes the channel is defined in the same file
- **kwargs** (*Any*) – named arguments for required channel density type

Returns
None

add_intracellular_property(*property_name*, ***kwargs*)

Generic function to add an intracellular property to the cell.

For a full list of membrane properties, see: https://docs.neuroml.org/Userdocs/Schemas/Cells.html?#intracellularproperties

Parameters

- **property_name** (*str*) – name of intracellular property to add
- **kwargs** (*Any*) – named arguments for intracellular property to be added

Returns
None

add_membrane_property(*property_name*, ***kwargs*)

Generic function to add a membrane property to the cell.

For a full list of membrane properties, see: https://docs.neuroml.org/Userdocs/Schemas/Cells.html?#membraneproperties

Please also see specific functions in this module, which are designed to be easier to use than this generic function.

Parameters

- **property_name** (*str*) – name of membrane to add
- **kwargs** (*Any*) – named arguments for membrane property to be added

Returns
None

add_segment(*prox*, *dist*, *seg_id=None*, *name=None*, *parent=None*, *fraction_along=1.0*, *group_id=None*, *use_convention=True*, *seg_type=None*, *reorder_segment_groups=True*)

Add a segment to the cell, to the provided segment group, creating it if required.

Parameters

- **prox** (*list with 4 float entries: [x, y, z, diameter]*) – proximal segment information
- **dist** (*list with 4 float entries: [x, y, z, diameter]*) – dist segment information
- **seg_id** (*str*) – explicit ID to set for segment When not provided, the function will automatically add an ID based on the number of segments already included in the cell. It is best to either always set an explicit ID or let the function set it automatically, but not to mix the two. A *ValueError* is raised if a segment with the provided ID already exists
- **name** (*str*) – name of segment If a name is given, it is used. If no name is given, but a segment group is provided, the segment is named: "Seg<number>_<group name>" where <number> is the number of the segment in the segment group. (to be read as "segment <number> in <group>"; the group name should indicate the type here) If no name is given, and no segment group is provided, the segment is simply named: "Seg<segment id>".
- **parent** (SegmentParent) – parent segment
- **fraction_along** (*float*) – where the new segment is connected to the parent (0: distal point, 1: proximal point)
- **group_id** (*str*) – id of segment group to add the segment to If a segment group with this id does not exist, a new segment group will be created.

The suggested convention is: *axon_*, *soma_*, *dend_* for axonal, somatic, and dendritic segment groups respectively.

Note that a newly created segment group will not be marked as an unbranched segment group. If you wish to add a segment to an unbranched segment group, please create one using *add_unbranched_segment_group* and then add segments to it.

- **use_convention** (*bool*) – whether the segment or its group should be added to the global segment groups. The *seg_type* notes what global group this segment or its segment group should also be added to.
- **reorder_segment_groups** (*bool*) – whether the groups should be reordered to put the default segment groups last after the segment has been added. This is required for a valid NeuroML file because segment groups included in the default groups should be declared before they are used in the default groups. When adding lots of segments, one may want to only reorder at the end of the process instead of after each segment is added.

  This is only relevant if *use_convention=True*.

Seg_type
type of segment ("axon", "dendrite", "soma") If *use_convention* is *True*, and a *group_id* is provided, the segment group will also be added to the default segment groups if it has not been previously added. If *group_id* is *None*, the segment will be added to the default groups instead.

If *use_convention* is *False*, this is unused.

Returns
the created segment

Return type
Segment

Raises
ValueError – if *seg_id* is provided and a segment with this ID already exists

add_segment_group(*group_id*)

Add a new general segment group.

The segments included in this group do not need to be contiguous. This segment group will not be marked as a section using the required NeuroLex ID.

Parameters
group_id (*str*) – ID of segment group

Returns
new segment group

Return type
SegmentGroup

add_unbranched_segment_group(*group_id*)

Add a new unbranched segment group.

This is similar to the *add_segment_group* method, but this segment group will be used to store contiguous segments, which form an unbranched section of a cell.

Parameters
group_id (*str*) – ID of segment group

Returns
new segment group

Return type
SegmentGroup

add_unbranched_segments(*points*, *parent=None*, *fraction_along=1.0*, *group_id=None*, *use_convention=True*, *seg_type=None*)

Add an unbranched list of segments to the cell.

The list of points will include the first proximal point where this should be joined to the cell, followed by a list of distal points:

```
|-----|-----|-----|------|.....---|
p1    d1    d2    d3     d4       d N-1
```

So, a list of N points will create a list of N-1 segments

The list of points will be of the form:

```
[[x1, y1, z1, d1], [x2, y2, z2, d2] ...]
```

Please ensure that the first point, p1, is correctly set to ensure that this segment list is correctly connected to the rest of the cell.

Parameters

- **points** (*list of [x, y, z, d] points*) – 3D points to create the segments
- **parent** (SegmentParent) – parent segment where first segment of list is to be attached
- **fraction_along** (*float*) – where the new segment list is connected to the parent (0: distal point, 1: proximal point) Note that the second and following segments will all be added at the distal point of the previous segment
- **group_id** (SegmentGroup) – segment group to add the segment to if a segment group does not already exist, it will be created
- **use_convention** (*bool*) – whether helper segment groups should be created using the default convention See the documentation of the *add_segment* method for more information on the convention
- **seg_type** (*str*) – type of segments ("axon", "soma", "dendrite")

Returns
the segment group containing this new list of segments

Return type
SegmentGroup

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

create_unbranched_segment_group_branches(*root_segment_id: int*, *use_convention: bool = True*)

Organise the segments of the cell into new segment groups that each form a single contiguous unbranched cell branch.

Note that the first segment (root segment) of a branch must have a proximal point that connects it to the rest of the neuronal morphology. If, when constructing these branches, a root segment is found that does not include a proximal point, one will be added using the *get_actual_proximal* method.

No other changes will be made to any segments, or to any pre-existing segment groups.

Parameters

- **root_segment_id** (*int*) – id of segment considered the root of the tree, generally the first soma segment
- **use_convention** (*bool*) – toggle using NeuroML convention for segment groups

Returns
modified cell with new section groups

Return type
neuroml.Cell

get_actual_proximal(*segment_id*)

Get the proximal point of a segment.

If the proximal for the segment is set to None, calculate the proximal on the parent using fraction_along and return it.

Parameters
segment_id – ID of segment

Returns
proximal point

get_all_segments_in_group(*segment_group*, *assume_all_means_all=True*)

Get all the segments in a segment group of the cell.

Parameters

- **segment_group** – segment group to get all segments of
- **assume_all_means_all** – return all segments if the "all" segment group wasn't explicitly defined

Returns
list of segment ids

Return type
list[int]

Raises
Exception – if no segment group is found in the cell.

get_ordered_segments_in_groups(*group_list*, *check_parentage=False*, *include_cumulative_lengths=False*, *include_path_lengths=False*, *path_length_metric='Path Length from root'*)

Get ordered list of segments in specified groups

Parameters

- **group_list** (*str or list*) – a group id or list of groups to get segments from
- **check_parentage** (*bool*) – verify parentage
- **include_commulative_lengths** – also include cummulative lengths
- **include_path_lengths** (*bool*) – also include path lengths
- **path_length_metric** (*str*) – metric to use for path length ("Path Length from root" is currently the only supported option, and the default)

Returns
dictionary of segments with additional information depending on what parameters were used:

Raises
Exception if check_parentage is True and parentage cannot be verified

get_segment(*segment_id*)

Get segment object by its id

Parameters
segment_id – ID of segment

Returns
segment

Raises
ValueError – if the segment is not found in the cell

get_segment_adjacency_list()

Get the adjacency list of all segments in the cell morphology. Returns a dict where each key is a parent segment, and the value is the list of its children segments.

Segment without children (leaf segments) are not included as parents in the adjacency list.

Returns
dict with parent segments as keys and their children as values

Return type
dict

`get_segment_group`(*sg_id*)

Return the SegmentGroup object for the specified segment group id.

Parameters
`sg_id` (*str*) – id of segment group to find

Returns
SegmentGroup object of specified ID

Raises
`ValueError` – if segment group is not found in cell

`get_segment_groups_by_substring`(*substring*)

Get a dictionary of segment group IDs and the segment groups matching the specified substring

Parameters
`substring` (*str*) – substring to match

Returns
dictionary with segment group ID as key, and segment group as value

Raises
`ValueError` – if no matching segment groups are found in cell

`get_segment_ids_vs_segments`()

Get a dictionary of segment IDs and the segments in the cell.

Returns
dictionary with segment ID as key, and segment as value

`get_segment_length`(*segment_id*)

Get the length of the segment.

Parameters
`segment_id` – ID of segment

Returns
length of segment

`get_segment_surface_area`(*segment_id*)

Get the surface area of the segment.

Parameters
`segment_id` – ID of the segment

Returns
surface area of segment

`get_segment_volume`(*segment_id*)

Get volume of segment

Parameters
`segment_id` – ID of the segment

Returns
volume of the segment

`get_segments_by_substring`(*substring*)

Get a dictionary of segment IDs and the segment matching the specified substring

Parameters
`substring` (*str*) – substring to match

Returns
dictionary with segment ID as key, and segment as value

Raises
Exception – if no segments are found

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

neuro_lex_ids = {'axon': 'GO:0030424', 'dend': 'GO:0030425', 'section': 'sao864921383', 'soma': 'GO:0043025'}

optimise_segment_group(*seg_group_id*)

Optimise segment group with id *seg_group_id*.

Parameters
seg_group_id (*str*) – id of segment group to optimise

optimise_segment_groups()

Optimise all segment groups in the cell.

This will:

- deduplicate members and includes in segment groups
- remove members that have already been included using a segment group

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

reorder_segment_groups()

Move default segment groups to the end.

This is required so that the segment groups included in the default groups are defined before they are used.

Returns
None

set_init_memb_potential(*v*, *group_id='all'*)

Set the initial membrane potential of the cell.

Parameters

- **v** (*str*) – value to set for membrane potential with units
- **group_id** (*str*) – id of segment group to modify

set_resistivity(*resistivity*, *group_id='all'*) → None

Set the resistivity of the cell

Parameters
group_id (*str*) – segment group to modify

set_specific_capacitance(*spec_cap*, *group_id='all'*)

Set the specific capacitance for the cell.

Parameters

- **spec_cap** (*str*) – value of specific capacitance with units
- **group_id** (*str*) – segment group to modify

`set_spike_thresh`(*v*, *group_id='all'*)

Set the spike threshold of the cell.

Parameters

- **v** (*str*) – value to set for spike threshold with units
- **group_id** (*str*) – id of segment group to modify

`setup_nml_cell`(*use_convention=True*, *overwrite=False*)

Correctly initialise a NeuroML cell.

To be called after a new component has been created to initialise the cell with these properties:

- Morphology: id="morphology"
- BiophysicalProperties: id="biophys":
  - MembraneProperties
  - IntracellularProperties

If *use_convention* is True, it also creates some default SegmentGroups for convenience:

- "all", "soma_group", "dendrite_group", "axon_group" which are used by other helper functions to include all, soma, dendrite, and axon segments respectively.

Note that since this cell does not currently include a segment in its morphology, it is *not* a valid NeuroML construct. Use the *add_segment* and *add_unbranched_segments* functions to add segments and branches. They will also populate the default segment groups.

Parameters

- **id** (*str*) – id of the cell
- **use_convention** (*bool*) – whether helper segment groups should be created using the default convention
- **overwrite** (*bool*) – overwrite existing components

Returns
None

Return type
None

`summary`()

Print cell summary.

Currently prints:

- id of cell
- any notes
- number of segments
- number of segment groups

TODO: extend to show more information about the cell that may be useful to users.

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Cell2CaPools

class neuroml.nml.nml.**Cell2CaPools**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *morphology_attr: a NmlId (optional) = None*, *biophysical_properties_attr: a NmlId (optional) = None*, *morphology: a Morphology (optional) = None*, *biophysical_properties: a BiophysicalProperties (optional) = None*, *biophysical_properties2_ca_pools: a BiophysicalProperties2CaPools (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Cell*

Cell2CaPools – Variant of cell with two independent Ca2+ pools. Cell with **segment** s specified in a **morphology** element along with details on its **biophysicalProperties** . NOTE: this can only be correctly simulated using jLEMS when there is a single segment in the cell, and **v** of this cell represents the membrane potential in that isopotential segment.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

add_channel_density(*nml_cell_doc*, *cd_id*, *ion_channel*, *cond_density*, *erev='0.0 mV'*, *group_id='all'*, *ion='non_specific'*, *ion_chan_def_file=''*)

Add channel density.

Parameters

- **nml_cell_doc** (NeuroMLDocument) – cell NeuroML document to which channel density is to be added
- **cd_id** (*str*) – id for channel density
- **ion_channel** (*str*) – name of ion channel
- **cond_density** (*str*) – value of conductance density with units
- **erev** (*str*) – value of reversal potential with units
- **group_id** (*str*) – segment groups to add to
- **ion** (*str*) – name of ion
- **ion_chan_def_file** (*str*) – path to NeuroML2 file defining the ion channel, if empty, it assumes the channel is defined in the same file

add_channel_density_v(*channel_density_type*, *nml_cell_doc*, *ion_chan_def_file=''*, ***kwargs*)

Generic function to add channel density components to a Cell.

Parameters

- **channel_density_type** (*str*) – type of channel density to add. See https://docs.neuroml.org/Userdocs/Schemas/Cells.html for the complete list.
- **nml_cell_doc** (NeuroMLDocument) – cell NeuroML document to which channel density is to be added
- **ion_chan_def_file** (*str*) – path to NeuroML2 file defining the ion channel, if empty, it assumes the channel is defined in the same file
- **kwargs** (*Any*) – named arguments for required channel density type

Returns
None

add_intracellular_property(*property_name*, ***kwargs*)

Generic function to add an intracellular property to the cell.

For a full list of membrane properties, see: https://docs.neuroml.org/Userdocs/Schemas/Cells.html?#intracellularproperties

Parameters

- **property_name** (*str*) – name of intracellular property to add
- **kwargs** (*Any*) – named arguments for intracellular property to be added

Returns
None

add_membrane_property(*property_name*, ***kwargs*)

Generic function to add a membrane property to the cell.

For a full list of membrane properties, see: https://docs.neuroml.org/Userdocs/Schemas/Cells.html?#membraneproperties

Please also see specific functions in this module, which are designed to be easier to use than this generic function.

Parameters

- **property_name** (*str*) – name of membrane to add
- **kwargs** (*Any*) – named arguments for membrane property to be added

Returns
None

add_segment(*prox*, *dist*, *seg_id=None*, *name=None*, *parent=None*, *fraction_along=1.0*, *group_id=None*, *use_convention=True*, *seg_type=None*, *reorder_segment_groups=True*)

Add a segment to the cell, to the provided segment group, creating it if required.

Parameters

- **prox** (*list with 4 float entries: [x, y, z, diameter]*) – proximal segment information
- **dist** (*list with 4 float entries: [x, y, z, diameter]*) – dist segment information
- **seg_id** (*str*) – explicit ID to set for segment When not provided, the function will automatically add an ID based on the number of segments already included in the cell. It is best to either always set an explicit ID or let the function set it automatically, but not to mix the two. A *ValueError* is raised if a segment with the provided ID already exists
- **name** (*str*) – name of segment If a name is given, it is used. If no name is given, but a segment group is provided, the segment is named: "Seg<number>_<group name>" where <number> is the number of the segment in the segment group. (to be read as "segment <number> in <group>"; the group name should indicate the type here) If no name is given, and no segment group is provided, the segment is simply named: "Seg<segment id>".
- **parent** (SegmentParent) – parent segment
- **fraction_along** (*float*) – where the new segment is connected to the parent (0: distal point, 1: proximal point)
- **group_id** (*str*) – id of segment group to add the segment to If a segment group with this id does not exist, a new segment group will be created.

  The suggested convention is: *axon_*, *soma_*, *dend_* for axonal, somatic, and dendritic segment groups respectively.

  Note that a newly created segment group will not be marked as an unbranched segment group. If you wish to add a segment to an unbranched segment group, please create one using *add_unbranched_segment_group* and then add segments to it.
- **use_convention** (*bool*) – whether the segment or its group should be added to the global segment groups. The *seg_type* notes what global group this segment or its segment group should also be added to.
- **reorder_segment_groups** (*bool*) – whether the groups should be reordered to put the default segment groups last after the segment has been added. This is required for a valid NeuroML file because segment groups included in the default groups should be declared

before they are used in the default groups. When adding lots of segments, one may want to only reorder at the end of the process instead of after each segment is added.

This is only relevant if *use_convention=True*.

Seg_type
type of segment ("axon", "dendrite", "soma") If *use_convention* is *True*, and a *group_id* is provided, the segment group will also be added to the default segment groups if it has not been previously added. If *group_id* is *None*, the segment will be added to the default groups instead.

If *use_convention* is *False*, this is unused.

Returns
the created segment

Return type
Segment

Raises
ValueError – if *seg_id* is provided and a segment with this ID already exists

add_segment_group(*group_id*)

Add a new general segment group.

The segments included in this group do not need to be contiguous. This segment group will not be marked as a section using the required NeuroLex ID.

Parameters
group_id (*str*) – ID of segment group

Returns
new segment group

Return type
SegmentGroup

add_unbranched_segment_group(*group_id*)

Add a new unbranched segment group.

This is similar to the *add_segment_group* method, but this segment group will be used to store contiguous segments, which form an unbranched section of a cell.

Parameters
group_id (*str*) – ID of segment group

Returns
new segment group

Return type
SegmentGroup

add_unbranched_segments(*points*, *parent=None*, *fraction_along=1.0*, *group_id=None*, *use_convention=True*, *seg_type=None*)

Add an unbranched list of segments to the cell.

The list of points will include the first proximal point where this should be joined to the cell, followed by a list of distal points:

```
|-----|-----|-----|------|.....---|
p1    d1    d2    d3     d4       d N-1
```

So, a list of N points will create a list of N-1 segments

The list of points will be of the form:

```
[[x1, y1, z1, d1], [x2, y2, z2, d2] ...]
```

Please ensure that the first point, p1, is correctly set to ensure that this segment list is correctly connected to the rest of the cell.

Parameters

- **points** (*list of [x, y, z, d] points*) – 3D points to create the segments
- **parent** (SegmentParent) – parent segment where first segment of list is to be attached
- **fraction_along** (*float*) – where the new segment list is connected to the parent (0: distal point, 1: proximal point) Note that the second and following segments will all be added at the distal point of the previous segment
- **group_id** (SegmentGroup) – segment group to add the segment to if a segment group does not already exist, it will be created
- **use_convention** (*bool*) – whether helper segment groups should be created using the default convention See the documentation of the *add_segment* method for more information on the convention
- **seg_type** (*str*) – type of segments ("axon", "soma", "dendrite")

Returns
the segment group containing this new list of segments

Return type
SegmentGroup

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

create_unbranched_segment_group_branches(*root_segment_id: int*, *use_convention: bool = True*)

Organise the segments of the cell into new segment groups that each form a single contiguous unbranched cell branch.

Note that the first segment (root segment) of a branch must have a proximal point that connects it to the rest of the neuronal morphology. If, when constructing these branches, a root segment is found that does not include a proximal point, one will be added using the *get_actual_proximal* method.

No other changes will be made to any segments, or to any pre-existing segment groups.

Parameters

- **root_segment_id** (*int*) – id of segment considered the root of the tree, generally the first soma segment
- **use_convention** (*bool*) – toggle using NeuroML convention for segment groups

Returns
modified cell with new section groups

Return type
neuroml.Cell

get_actual_proximal(*segment_id*)

Get the proximal point of a segment.

If the proximal for the segment is set to None, calculate the proximal on the parent using fraction_along and return it.

Parameters
segment_id – ID of segment

Returns
proximal point

get_all_segments_in_group(*segment_group*, *assume_all_means_all=True*)

Get all the segments in a segment group of the cell.

Parameters

- **segment_group** – segment group to get all segments of
- **assume_all_means_all** – return all segments if the "all" segment group wasn't explicitly defined

Returns
list of segment ids

Return type
list[int]

Raises
Exception – if no segment group is found in the cell.

get_ordered_segments_in_groups(*group_list*, *check_parentage=False*, *include_cumulative_lengths=False*, *include_path_lengths=False*, *path_length_metric='Path Length from root'*)

Get ordered list of segments in specified groups

Parameters

- **group_list** (*str or list*) – a group id or list of groups to get segments from
- **check_parentage** (*bool*) – verify parentage
- **include_commulative_lengths** – also include cummulative lengths
- **include_path_lengths** (*bool*) – also include path lengths
- **path_length_metric** (*str*) – metric to use for path length ("Path Length from root" is currently the only supported option, and the default)

Returns
dictionary of segments with additional information depending on what parameters were used:

Raises
Exception if check_parentage is True and parentage cannot be verified

get_segment(*segment_id*)

Get segment object by its id

Parameters
segment_id – ID of segment

Returns
segment

Raises
ValueError – if the segment is not found in the cell

get_segment_adjacency_list()

Get the adjacency list of all segments in the cell morphology. Returns a dict where each key is a parent segment, and the value is the list of its children segments.

Segment without children (leaf segments) are not included as parents in the adjacency list.

Returns
dict with parent segments as keys and their children as values

Return type
dict

get_segment_group(*sg_id*)

Return the SegmentGroup object for the specified segment group id.

Parameters
sg_id (*str*) – id of segment group to find

Returns
SegmentGroup object of specified ID

Raises
ValueError – if segment group is not found in cell

get_segment_groups_by_substring(*substring*)

Get a dictionary of segment group IDs and the segment groups matching the specified substring

Parameters
substring (*str*) – substring to match

Returns
dictionary with segment group ID as key, and segment group as value

Raises
ValueError – if no matching segment groups are found in cell

get_segment_ids_vs_segments()

Get a dictionary of segment IDs and the segments in the cell.

Returns
dictionary with segment ID as key, and segment as value

get_segment_length(*segment_id*)

Get the length of the segment.

Parameters
segment_id – ID of segment

Returns
length of segment

get_segment_surface_area(*segment_id*)

Get the surface area of the segment.

Parameters
segment_id – ID of the segment

Returns
surface area of segment

get_segment_volume(*segment_id*)

Get volume of segment

Parameters
segment_id – ID of the segment

Returns
volume of the segment

get_segments_by_substring(*substring*)

Get a dictionary of segment IDs and the segment matching the specified substring

Parameters
substring (*str*) – substring to match

Returns
dictionary with segment ID as key, and segment as value

Raises
Exception – if no segments are found

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

```
neuro_lex_ids = {'axon':  'GO:0030424', 'dend':  'GO:0030425', 'section':
'sao864921383', 'soma':  'GO:0043025'}
```

optimise_segment_group(*seg_group_id*)

Optimise segment group with id *seg_group_id*.

Parameters
seg_group_id (*str*) – id of segment group to optimise

optimise_segment_groups()

Optimise all segment groups in the cell.

This will:

- deduplicate members and includes in segment groups
- remove members that have already been included using a segment group

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids

can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

reorder_segment_groups()

Move default segment groups to the end.

This is required so that the segment groups included in the default groups are defined before they are used.

Returns
None

set_init_memb_potential(*v*, *group_id='all'*)

Set the initial membrane potential of the cell.

Parameters

- **v** (*str*) – value to set for membrane potential with units
- **group_id** (*str*) – id of segment group to modify

set_resistivity(*resistivity*, *group_id='all'*) → None

Set the resistivity of the cell

Parameters
group_id (*str*) – segment group to modify

set_specific_capacitance(*spec_cap*, *group_id='all'*)

Set the specific capacitance for the cell.

Parameters

- **spec_cap** (*str*) – value of specific capacitance with units
- **group_id** (*str*) – segment group to modify

set_spike_thresh(*v*, *group_id='all'*)

Set the spike threshold of the cell.

Parameters

- **v** (*str*) – value to set for spike threshold with units
- **group_id** (*str*) – id of segment group to modify

setup_nml_cell(*use_convention=True*, *overwrite=False*)

Correctly initialise a NeuroML cell.

To be called after a new component has been created to initialise the cell with these properties:

- Morphology: id="morphology"
- BiophysicalProperties: id="biophys":

- MembraneProperties
- IntracellularProperties

If *use_convention* is True, it also creates some default SegmentGroups for convenience:

- "all", "soma_group", "dendrite_group", "axon_group" which are used by other helper functions to include all, soma, dendrite, and axon segments respectively.

Note that since this cell does not currently include a segment in its morphology, it is *not* a valid NeuroML construct. Use the *add_segment* and *add_unbranched_segments* functions to add segments and branches. They will also populate the default segment groups.

Parameters

- **id** (*str*) – id of the cell
- **use_convention** (*bool*) – whether helper segment groups should be created using the default convention
- **overwrite** (*bool*) – overwrite existing components

Returns
None

Return type
None

summary()

Print cell summary.

Currently prints:

- id of cell
- any notes
- number of segments
- number of segment groups

TODO: extend to show more information about the cell that may be useful to users.

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

CellSet

class neuroml.nml.nml.**CellSet**(*id: a NmlId (required) = None*, *select: a string (required) = None*, *anytypeobjs_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ChannelDensity

class neuroml.nml.nml.**ChannelDensity**(*id: a NmlId (required) = None*, *ion_channel: a NmlId (required) = None*, *cond_density: a Nml2Quantity_conductanceDensity (optional) = None*, *erev: a Nml2Quantity_voltage (required) = None*, *segment_groups: a NmlId (optional) = 'all'*, *segments: a NonNegativeInteger (optional) = None*, *ion: a NmlId (required) = None*, *variable_parameters: list of VariableParameter(s) (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

ChannelDensity – Specifies a time varying ohmic conductance density, **gDensity,** which is distributed on an area of the **cell** (specified in **membraneProperties**) with fixed reversal potential **erev** producing a current density **iDensity**

Parameters

- **erev** (*voltage*) – The reversal potential of the current produced
- **condDensity** (*conductanceDensity*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ChannelDensityGHK

class neuroml.nml.nml.**ChannelDensityGHK**(*id: a NmlId (required) = None*, *ion_channel: a NmlId (required) = None*, *permeability: a Nml2Quantity_permeability (required) = None*, *segment_groups: a NmlId (optional) = 'all'*, *segments: a NonNegativeInteger (optional) = None*, *ion: a NmlId (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

ChannelDensityGHK – Specifies a time varying conductance density, **gDensity,** which is distributed on an area of the cell, producing a current density **iDensity** and whose reversal potential is calculated from the Goldman Hodgkin Katz equation. Hard coded for Ca only! See https://github.com/OpenSourceBrain/ghk-nernst.

Parameters
permeability (*permeability*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters

return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ChannelDensityGHK2

`class` neuroml.nml.nml.`ChannelDensityGHK2`(*id: a NmlId (required) = None*, *ion_channel: a NmlId (required) = None*, *cond_density: a Nml2Quantity_conductanceDensity (optional) = None*, *segment_groups: a NmlId (optional) = 'all'*, *segments: a NonNegativeInteger (optional) = None*, *ion: a NmlId (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

ChannelDensityGHK2 – Time varying conductance density, **gDensity,** which is distributed on an area of the cel l, producing a current density **iDensity.** Modified version of Jaffe et al. 1994 (used also in Lawrence et al. 2006). See https://github.com/OpenSourceBrain/ghk-nernst.

Parameters
condDensity (*conductanceDensity*) –

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None

- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ChannelDensityNernst

class neuroml.nml.nml.**ChannelDensityNernst**(*id: a NmlId (required) = None*, *ion_channel: a NmlId (required) = None*, *cond_density: a Nml2Quantity_conductanceDensity (optional) = None*, *segment_groups: a NmlId (optional) = 'all'*, *segments: a NonNegativeInteger (optional) = None*, *ion: a NmlId (required) = None*, *variable_parameters: list of VariableParameter(s) (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

ChannelDensityNernst – Specifies a time varying conductance density, **gDensity,** which is distributed on an area of the **cell,** producing a current density **iDensity** and whose reversal potential is calculated from the Nernst equation. Hard coded for Ca only! See https://github.com/OpenSourceBrain/ghk-nernst.

Parameters
condDensity (*conductanceDensity*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously

- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ChannelDensityNernstCa2

class neuroml.nml.nml.**ChannelDensityNernstCa2**(*id: a NmlId (required) = None*, *ion_channel: a NmlId (required) = None*, *cond_density: a Nml2Quantity_conductanceDensity (optional) = None*, *segment_groups: a NmlId (optional) = 'all'*, *segments: a NonNegativeInteger (optional) = None*, *ion: a NmlId (required) = None*, *variable_parameters: list of VariableParameter(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *ChannelDensityNernst*

ChannelDensityNernstCa2 – This component is similar to the original component type **channelDensityNernst** but it is changed in order to have a reversal potential that depends on a second independent Ca++ pool (ca2). See https://github.com/OpenSourceBrain/ghk-nernst.

Parameters
condDensity (*conductanceDensity*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ChannelDensityNonUniform

class neuroml.nml.nml.**ChannelDensityNonUniform**(*id: a NmlId (required) = None*, *ion_channel: a NmlId (required) = None*, *erev: a Nml2Quantity_voltage (required) = None*, *ion: a NmlId (required) = None*, *variable_parameters: list of VariableParameter(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

ChannelDensityNonUniform – Specifies a time varying ohmic conductance density, which is distributed on a region of the **cell.** The conductance density of the channel is not uniform, but is set using the **variableParameter** . Note, there is no dynamical description of this in LEMS yet, as this type only makes sense for multicompartmental cells. A ComponentType for this needs to be present to enable export of NeuroML 2 multicompartmental cells via LEMS/jNeuroML to NEURON

Parameters
erev (*voltage*) – The reversal potential of the current produced

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided,

only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
`return_format` (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
`recursive` (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
`ValueError` – if component is invalid

ChannelDensityNonUniformGHK

class neuroml.nml.nml.**ChannelDensityNonUniformGHK**(*id: a NmlId (required) = None*, *ion_channel: a NmlId (required) = None*, *ion: a NmlId (required) = None*, *variable_parameters: list of VariableParameter(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

ChannelDensityNonUniformGHK – Specifies a time varying conductance density, which is distributed on a region of the **cell,** and whose current is calculated from the Goldman-Hodgkin-Katz equation. Hard coded for Ca only!. The conductance density of the channel is not uniform, but is set using the **variableParameter** . Note, there is no dynamical description of this in LEMS yet, as this type only makes sense for multicompartmental cells. A ComponentType for this needs to be present to enable export of NeuroML 2 multicompartmental cells via LEMS/jNeuroML to NEURON

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ChannelDensityNonUniformNernst

class neuroml.nml.nml.**ChannelDensityNonUniformNernst**(*id: a NmlId (required) = None*, *ion_channel: a NmlId (required) = None*, *ion: a NmlId (required) = None*, *variable_parameters: list of VariableParameter(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

ChannelDensityNonUniformNernst – Specifies a time varying conductance density, which is distributed on a region of the **cell,** and whose reversal potential is calculated from the Nernst equation. Hard coded for Ca only!. The conductance density of the channel is not uniform, but is set using the **variableParameter** . Note, there is no dynamical description of this in LEMS yet, as this type only makes sense for multicompartmental cells. A ComponentType for this needs to be present to enable export of NeuroML 2 multicompartmental cells via LEMS/jNeuroML to NEURON

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ChannelDensityVShift

class neuroml.nml.nml.**ChannelDensityVShift**(*id: a NmlId (required) = None*, *ion_channel: a NmlId (required) = None*, *cond_density: a Nml2Quantity_conductanceDensity (optional) = None*, *erev: a Nml2Quantity_voltage (required) = None*, *segment_groups: a NmlId (optional) = 'all'*, *segments: a NonNegativeInteger (optional) = None*, *ion: a NmlId (required) = None*, *variable_parameters: list of VariableParameter(s) (optional) = None*, *v_shift: a Nml2Quantity_voltage (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *ChannelDensity*

ChannelDensityVShift – Same as **channelDensity** , but with a **vShift** parameter to change voltage activation of gates. The exact usage of **vShift** in expressions for rates is determined by the individual gates.

Parameters

- **vShift** (*voltage*) –
- **erev** (*voltage*) – The reversal potential of the current produced
- **condDensity** (*conductanceDensity*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ChannelPopulation

class neuroml.nml.nml.**ChannelPopulation**(*id: a NmlId (required) = None*, *ion_channel: a NmlId (required) = None*, *number: a NonNegativeInteger (required) = None*, *erev: a Nml2Quantity_voltage (required) = None*, *segment_groups: a NmlId (optional) = 'all'*, *segments: a NonNegativeInteger (optional) = None*, *ion: a NmlId (required) = None*, *variable_parameters: list of VariableParameter(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

ChannelPopulation – Population of a **number** of ohmic ion channels. These each produce a conductance **channelg** across a reversal potential **erev,** giving a total current **i.** Note that active membrane currents are more frequently specified as a density over an area of the **cell** using **channelDensity**

Parameters

- **number** (*none*) – The number of channels present. This will be multiplied by the time varying conductance of the individual ion channel (which extends **baseIonChannel**) to produce the total conductance
- **erev** (*voltage*) – The reversal potential of the current produced

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ClosedState

class neuroml.nml.nml.**ClosedState**(*id: a NmlId (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

ClosedState – A **KSState** with **relativeConductance** of 0

Parameters
relativeConductance (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need

to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ComponentType

class neuroml.nml.nml.**ComponentType**(*name: a string (required) = None*, *extends: a string (optional) = None*, *description: a string (optional) = None*, *Property: list of Property(s) (optional) = None*, *Parameter: list of Parameter(s) (optional) = None*, *Constant: list of Constant(s) (optional) = None*, *Exposure: list of Exposure(s) (optional) = None*, *Requirement: list of Requirement(s) (optional) = None*, *InstanceRequirement: list of InstanceRequirement(s) (optional) = None*, *Dynamics: list of Dynamics(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *GeneratedsSuper*

ComponentType – Contains an extension to NeuroML by creating custom LEMS ComponentType.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

CompoundInput

class neuroml.nml.nml.**CompoundInput**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *pulse_generators: list of PulseGenerator(s) (optional) = None*, *sine_generators: list of SineGenerator(s) (optional) = None*, *ramp_generators: list of RampGenerator(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

CompoundInput – Generates a current which is the sum of all its child **basePointCurrent** element, e. g. can be a combination of **pulseGenerator** , **sineGenerator** elements producing a single **i.** Scaled by **weight,** if set

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child

elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

CompoundInputDL

class neuroml.nml.nml.**CompoundInputDL**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *pulse_generator_dls: list of PulseGeneratorDL(s) (optional) = None*, *sine_generator_dls: list of SineGeneratorDL(s) (optional) = None*, *ramp_generator_dls: list of RampGeneratorDL(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

CompoundInputDL – Generates a current which is the sum of all its child **basePointCurrentDL** elements, e. g. can be a combination of **pulseGeneratorDL** , **sineGeneratorDL** elements producing a single **i.** Scaled by **weight,** if set

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ConcentrationModel_D

class neuroml.nml.nml.**ConcentrationModel_D**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *ion: a NmlId (required) = None*, *resting_conc: a Nml2Quantity_concentration (required) = None*, *decay_constant: a Nml2Quantity_time (required) = None*, *shell_thickness: a Nml2Quantity_length (required) = None*, *type: a string (required) = 'decayingPoolConcentrationModel'*, *gds_collector_=None*, ***kwargs_*)

Bases: *DecayingPoolConcentrationModel*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ConditionalDerivedVariable

class neuroml.nml.nml.**ConditionalDerivedVariable**(*name: a string (required) = None*, *dimension: a string (required) = None*, *description: a string (optional) = None*, *exposure: a string (optional) = None*, *Case: list of Case(s) (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *NamedDimensionalVariable*

ConditionalDerivedVariable – LEMS ComponentType for ConditionalDerivedVariable

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Connection

class neuroml.nml.nml.**Connection**(*id: a NonNegativeInteger (required) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *pre_cell_id: a string (required) = None*, *pre_segment_id: a NonNegativeInteger (optional) = '0'*, *pre_fraction_along: a ZeroToOne (optional) = '0.5'*, *post_cell_id: a string (required) = None*, *post_segment_id: a NonNegativeInteger (optional) = '0'*, *post_fraction_along: a ZeroToOne (optional) = '0.5'*, *gds_collector_=None*, ***kwargs_*)

Bases: BaseConnectionOldFormat

Connection – Event connection directly between named components, which gets processed via a new instance of a **synapse** component which is created on the target component. Normally contained inside a **projection** element.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

get_post_cell_id()

Get the ID of the post-synaptic cell

Returns
ID of post-synaptic cell

Return type
str

get_post_fraction_along()

Get post-synaptic fraction along information

get_post_info()

Get post-synaptic information summary

get_post_segment_id()

Get the ID of the post-synpatic segment

Returns
ID of post-synaptic segment.

Return type
str

get_pre_cell_id()

Get the ID of the pre-synaptic cell

Returns
ID of pre-synaptic cell

Return type
str

get_pre_fraction_along()

Get pre-synaptic fraction along information

`get_pre_info()`

Get pre-synaptic information summary

`get_pre_segment_id()`

Get the ID of the pre-synpatic segment

Returns
ID of pre-synaptic segment.

Return type
str

`info(show_contents=False, return_format='string')`

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo(return_format='string')`

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ConnectionWD

class neuroml.nml.nml.**ConnectionWD**(*id: a NonNegativeInteger (required) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *pre_cell_id: a string (required) = None*, *pre_segment_id: a NonNegativeInteger (optional) = '0'*, *pre_fraction_along: a ZeroToOne (optional) = '0.5'*, *post_cell_id: a string (required) = None*, *post_segment_id: a NonNegativeInteger (optional) = '0'*, *post_fraction_along: a ZeroToOne (optional) = '0.5'*, *weight: a float (required) = None*, *delay: a Nml2Quantity_time (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseConnectionOldFormat*

ConnectionWD – Event connection between named components, which gets processed via a new instance of a synapse component which is created on the target component, includes setting of **weight** and **delay** for the synaptic connection

Parameters

- **weight** (*none*) –

- **delay** (*time*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

`get_delay_in_ms()`

Get connection delay in milli seconds

Returns
connection delay in milli seconds

Return type
float

`get_post_cell_id()`

Get the ID of the post-synaptic cell

Returns
ID of post-synaptic cell

Return type
str

`get_post_fraction_along()`

Get post-synaptic fraction along information

`get_post_info()`

Get post-synaptic information summary

`get_post_segment_id()`

Get the ID of the post-synpatic segment

Returns
ID of post-synaptic segment.

Return type
str

`get_pre_cell_id()`

Get the ID of the pre-synaptic cell

Returns
ID of pre-synaptic cell

Return type
str

`get_pre_fraction_along()`

Get pre-synaptic fraction along information

`get_pre_info()`

Get pre-synaptic information summary

`get_pre_segment_id()`

Get the ID of the pre-synpatic segment

Returns
ID of pre-synaptic segment.

Return type
str

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an

information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Constant

`class` neuroml.nml.nml.`Constant`(*name: a string (required) = None*, *dimension: a string (required) = None*, *value: a Nml2Quantity (required) = None*, *description: a string (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Constant – LEMS ComponentType for Constant.

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ContinuousConnection

class neuroml.nml.nml.**ContinuousConnection**(*id: a NonNegativeInteger (required) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *pre_cell: a string (required) = None*, *pre_segment: a NonNegativeInteger (optional) = '0'*, *pre_fraction_along: a ZeroToOne (optional) = '0.5'*, *post_cell: a string (required) = None*, *post_segment: a NonNegativeInteger (optional) = '0'*, *post_fraction_along: a ZeroToOne (optional) = '0.5'*, *pre_component: a NmlId (required) = None*, *post_component: a NmlId (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseConnectionNewFormat*

ContinuousConnection – An instance of a connection in a **continuousProjection** between **presynapticPopulation** to another **postsynapticPopulation** through a **preComponent** at the start and **postComponent** at the end. Can be used for analog synapses.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

get_post_cell_id()

Get the ID of the post-synaptic cell

Returns
ID of post-synaptic cell

Return type
str

get_post_fraction_along()

Get post-synaptic fraction along information

get_post_info()

Get post-synaptic information summary

get_post_segment_id()

Get the ID of the post-synpatic segment

Returns
ID of post-synaptic segment.

Return type
str

`get_pre_cell_id`()

Get the ID of the pre-synaptic cell

Returns
ID of pre-synaptic cell

Return type
str

`get_pre_fraction_along`()

Get pre-synaptic fraction along information

`get_pre_info`()

Get pre-synaptic information summary

`get_pre_segment_id`()

Get the ID of the pre-synpatic segment

Returns
ID of pre-synaptic segment.

Return type
str

`info`(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ContinuousConnectionInstance

class neuroml.nml.nml.**ContinuousConnectionInstance**(*id: a NonNegativeInteger (required) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *pre_cell: a string (required) = None*, *pre_segment: a NonNegativeInteger (optional) = '0'*, *pre_fraction_along: a ZeroToOne (optional) = '0.5'*, *post_cell: a string (required) = None*, *post_segment: a NonNegativeInteger (optional) = '0'*, *post_fraction_along: a ZeroToOne (optional) = '0.5'*, *pre_component: a NmlId (required) = None*, *post_component: a NmlId (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *ContinuousConnection*

ContinuousConnectionInstance – An instance of a connection in a **continuousProjection** between **presynapticPopulation** to another **postsynapticPopulation** through a **preComponent** at the start and **postComponent** at the end. Populations need to be of type **populationList** and contain **instance** and **location** elements. Can be used for analog synapses.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

`get_post_cell_id()`

Get the ID of the post-synaptic cell

Returns
ID of post-synaptic cell

Return type
str

`get_post_fraction_along()`

Get post-synaptic fraction along information

`get_post_info()`

Get post-synaptic information summary

`get_post_segment_id()`

Get the ID of the post-synpatic segment

Returns
ID of post-synaptic segment.

Return type
str

`get_pre_cell_id()`

Get the ID of the pre-synaptic cell

Returns
ID of pre-synaptic cell

Return type
str

get_pre_fraction_along()

Get pre-synaptic fraction along information

get_pre_info()

Get pre-synaptic information summary

get_pre_segment_id()

Get the ID of the pre-synpatic segment

Returns
ID of pre-synaptic segment.

Return type
str

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ContinuousConnectionInstanceW

class neuroml.nml.nml.**ContinuousConnectionInstanceW**(*id: a NonNegativeInteger (required) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *pre_cell: a string (required) = None*, *pre_segment: a NonNegativeInteger (optional) = '0'*, *pre_fraction_along: a ZeroToOne (optional) = '0.5'*, *post_cell: a string (required) = None*, *post_segment: a NonNegativeInteger (optional) = '0'*, *post_fraction_along: a ZeroToOne (optional) = '0.5'*, *pre_component: a NmlId (required) = None*, *post_component: a NmlId (required) = None*, *weight: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *ContinuousConnectionInstance*

ContinuousConnectionInstanceW – An instance of a connection in a **continuousProjection** between **presynapticPopulation** to another **postsynapticPopulation** through a **preComponent** at the start and **postComponent** at the end. Populations need to be of type **populationList** and contain **instance** and **location** elements. Can be used for analog synapses. Includes setting of **weight** for the connection

Parameters
weight (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class

type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

get_post_cell_id()

Get the ID of the post-synaptic cell

Returns
ID of post-synaptic cell

Return type
str

get_post_fraction_along()

Get post-synaptic fraction along information

get_post_info()

Get post-synaptic information summary

get_post_segment_id()

Get the ID of the post-synpatic segment

Returns
ID of post-synaptic segment.

Return type
str

get_pre_cell_id()

Get the ID of the pre-synaptic cell

Returns
ID of pre-synaptic cell

Return type
str

get_pre_fraction_along()

Get pre-synaptic fraction along information

get_pre_info()

Get pre-synaptic information summary

get_pre_segment_id()

Get the ID of the pre-synpatic segment

Returns
ID of pre-synaptic segment.

Return type
str

`get_weight`()

Get weight.

If weight is not set, the default value of 1.0 is returned.

`info`(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids

can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ContinuousProjection

class neuroml.nml.nml.**ContinuousProjection**(*id: a NmlId (required) = None*, *presynaptic_population: a NmlId (required) = None*, *postsynaptic_population: a NmlId (required) = None*, *continuous_connections: list of ContinuousConnection(s) (optional) = None*, *continuous_connection_instances: list of ContinuousConnectionInstance(s) (optional) = None*, *continuous_connection_instance_ws: list of ContinuousConnectionInstanceW(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseProjection*

ContinuousProjection – A projection between **presynapticPopulation** and **postsynapticPopulation** through components **preComponent** at the start and **postComponent** at the end of a **continuousConnection** or **continuousConnectionInstance** . Can be used for analog synapses.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

exportHdf5(*h5file*, *h5Group*)

Export to HDF5 file.

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters

return_format (*str*) – format in which to return information. If "string" (default), an

information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

DecayingPoolConcentrationModel

`class` neuroml.nml.nml.`DecayingPoolConcentrationModel`(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *ion: a NmlId (required) = None*, *resting_conc: a Nml2Quantity_concentration (required) = None*, *decay_constant: a Nml2Quantity_time (required) = None*, *shell_thickness: a Nml2Quantity_length (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

DecayingPoolConcentrationModel – Model of an intracellular buffering mechanism for **ion** (currently hard Coded to be calcium, due to requirement for **iCa**) which has a baseline level **restingConc** and tends to this value with time course **decayConstant.** The ion is assumed to occupy a shell inside the membrane of thickness **shellThickness.**

Parameters

- **restingConc** (*concentration*) –
- **decayConstant** (*time*) –
- **shellThickness** (*length*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

DerivedVariable

class neuroml.nml.nml.**DerivedVariable**(*name: a string (required) = None*, *dimension: a string (required) = None*, *description: a string (optional) = None*, *exposure: a string (optional) = None*, *value: a string (optional) = None*, *select: a string (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *NamedDimensionalVariable*

DerivedVariable – LEMS ComponentType for DerivedVariable

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to

- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

DistalDetails

class neuroml.nml.nml.**DistalDetails**(*normalization_end: a double (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

DoubleSynapse

class neuroml.nml.nml.**DoubleSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *synapse1: a NmlId (required) = None*, *synapse2: a NmlId (required) = None*, *synapse1_path: a string (required) = None*, *synapse2_path: a string (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseVoltageDepSynapse*

DoubleSynapse – Synapse consisting of two independent synaptic mechanisms (e. g. AMPA-R and NMDA-R), which can be easily colocated in connections

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Dynamics

`class` neuroml.nml.nml.`Dynamics`(*StateVariable: list of StateVariable(s) (optional) = None*, *DerivedVariable: list of DerivedVariable(s) (optional) = None*, *ConditionalDerivedVariable: list of ConditionalDerivedVariable(s) (optional) = None*, *TimeDerivative: list of TimeDerivative(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *GeneratedsSuper*

Dynamics – LEMS ComponentType for Dynamics

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

EIF_cond_alpha_isfa_ista

class neuroml.nml.nml.**EIF_cond_alpha_isfa_ista**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *cm: a float (required) = None*, *i_offset: a float (required) = None*, *tau_syn_E: a float (required) = None*, *tau_syn_I: a float (required) = None*, *v_init: a float (required) = None*, *tau_m: a float (required) = None*, *tau_refrac: a float (required) = None*, *v_reset: a float (required) = None*, *v_rest: a float (required) = None*, *v_thresh: a float (required) = None*, *e_rev_E: a float (required) = None*, *e_rev_I: a float (required) = None*, *a: a float (required) = None*, *b: a float (required) = None*, *delta_T: a float (required) = None*, *tau_w: a float (required) = None*, *v_spike: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *EIF_cond_exp_isfa_ista*

EIF_cond_alpha_isfa_ista – Adaptive exponential integrate and fire neuron according to Brette R and Gerstner W (2005) with alpha-function-shaped post-synaptic conductance

Parameters

- **v_spike** (*none*) –
- **delta_T** (*none*) –
- **tau_w** (*none*) –
- **a** (*none*) –
- **b** (*none*) –
- **e_rev_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **e_rev_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_refrac** (*none*) –
- **v_thresh** (*none*) –
- **tau_m** (*none*) –
- **v_rest** (*none*) –
- **v_reset** (*none*) –
- **cm** (*none*) –
- **i_offset** (*none*) –
- **tau_syn_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_syn_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **v_init** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found

- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

EIF_cond_exp_isfa_ista

class neuroml.nml.nml.**EIF_cond_exp_isfa_ista**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *cm: a float (required) = None*, *i_offset: a float (required) = None*, *tau_syn_E: a float (required) = None*, *tau_syn_I: a float (required) = None*, *v_init: a float (required) = None*, *tau_m: a float (required) = None*, *tau_refrac: a float (required) = None*, *v_reset: a float (required) = None*, *v_rest: a float (required) = None*, *v_thresh: a float (required) = None*, *e_rev_E: a float (required) = None*, *e_rev_I: a float (required) = None*, *a: a float (required) = None*, *b: a float (required) = None*, *delta_T: a float (required) = None*, *tau_w: a float (required) = None*, *v_spike: a float (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *basePyNNIaFCondCell*

EIF_cond_exp_isfa_ista – Adaptive exponential integrate and fire neuron according to Brette R and Gerstner W (2005) with exponentially-decaying post-synaptic conductance

Parameters

- **v_spike** (*none*) –
- **delta_T** (*none*) –
- **tau_w** (*none*) –
- **a** (*none*) –
- **b** (*none*) –
- **e_rev_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **e_rev_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_refrac** (*none*) –
- **v_thresh** (*none*) –
- **tau_m** (*none*) –
- **v_rest** (*none*) –
- **v_reset** (*none*) –
- **cm** (*none*) –
- **i_offset** (*none*) –

- **tau_syn_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_syn_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **v_init** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ElectricalConnection

class neuroml.nml.nml.**ElectricalConnection**(*id: a NonNegativeInteger (required) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *pre_cell: a string (required) = None*, *pre_segment: a NonNegativeInteger (optional) = '0'*, *pre_fraction_along: a ZeroToOne (optional) = '0.5'*, *post_cell: a string (required) = None*, *post_segment: a NonNegativeInteger (optional) = '0'*, *post_fraction_along: a ZeroToOne (optional) = '0.5'*, *synapse: a NmlId (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseConnectionNewFormat*

ElectricalConnection – To enable connections between populations through gap junctions.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

get_post_cell_id()

Get the ID of the post-synaptic cell

Returns
ID of post-synaptic cell

Return type
str

get_post_fraction_along()

Get post-synaptic fraction along information

get_post_info()

Get post-synaptic information summary

get_post_segment_id()

Get the ID of the post-synpatic segment

Returns
ID of post-synaptic segment.

Return type
str

get_pre_cell_id()

Get the ID of the pre-synaptic cell

Returns
ID of pre-synaptic cell

Return type
str

get_pre_fraction_along()

Get pre-synaptic fraction along information

get_pre_info()

Get pre-synaptic information summary

get_pre_segment_id()

Get the ID of the pre-synpatic segment

Returns
ID of pre-synaptic segment.

Return type
str

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ElectricalConnectionInstance

class neuroml.nml.nml.**ElectricalConnectionInstance**(*id: a NonNegativeInteger (required) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *pre_cell: a string (required) = None*, *pre_segment: a NonNegativeInteger (optional) = '0'*, *pre_fraction_along: a ZeroToOne (optional) = '0.5'*, *post_cell: a string (required) = None*, *post_segment: a NonNegativeInteger (optional) = '0'*, *post_fraction_along: a ZeroToOne (optional) = '0.5'*, *synapse: a NmlId (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *ElectricalConnection*

ElectricalConnectionInstance – To enable connections between populations through gap junctions. Populations need to be of type **populationList** and contain **instance** and **location** elements.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

get_post_cell_id()

Get the ID of the post-synaptic cell

Returns
ID of post-synaptic cell

Return type
str

get_post_fraction_along()

Get post-synaptic fraction along information

get_post_info()

Get post-synaptic information summary

`get_post_segment_id`()

Get the ID of the post-synpatic segment

Returns
ID of post-synaptic segment.

Return type
str

`get_pre_cell_id`()

Get the ID of the pre-synaptic cell

Returns
ID of pre-synaptic cell

Return type
str

`get_pre_fraction_along`()

Get pre-synaptic fraction along information

`get_pre_info`()

Get pre-synaptic information summary

`get_pre_segment_id`()

Get the ID of the pre-synpatic segment

Returns
ID of pre-synaptic segment.

Return type
str

`info`(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided,

only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ElectricalConnectionInstanceW

class neuroml.nml.nml.**ElectricalConnectionInstanceW**(*id: a NonNegativeInteger (required) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *pre_cell: a string (required) = None*, *pre_segment: a NonNegativeInteger (optional) = '0'*, *pre_fraction_along: a ZeroToOne (optional) = '0.5'*, *post_cell: a string (required) = None*, *post_segment: a NonNegativeInteger (optional) = '0'*, *post_fraction_along: a ZeroToOne (optional) = '0.5'*, *synapse: a NmlId (required) = None*, *weight: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *ElectricalConnectionInstance*

ElectricalConnectionInstanceW – To enable connections between populations through gap junctions. Populations need to be of type **populationList** and contain **instance** and **location** elements. Includes setting of **weight** for the connection

Parameters
weight (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

`get_post_cell_id()`

Get the ID of the post-synaptic cell

Returns
ID of post-synaptic cell

Return type
str

`get_post_fraction_along()`

Get post-synaptic fraction along information

`get_post_info()`

Get post-synaptic information summary

`get_post_segment_id()`

Get the ID of the post-synpatic segment

Returns
ID of post-synaptic segment.

Return type
str

`get_pre_cell_id()`

Get the ID of the pre-synaptic cell

Returns
ID of pre-synaptic cell

Return type
str

`get_pre_fraction_along`()

Get pre-synaptic fraction along information

`get_pre_info`()

Get pre-synaptic information summary

`get_pre_segment_id`()

Get the ID of the pre-synpatic segment

Returns
ID of pre-synaptic segment.

Return type
str

`get_weight`()

Get the weight of the connection

If a weight is not set (or is set to None), returns the default value of 1.0.

Returns
weight of connection or 1.0 if not set

Return type
float

`info`(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ElectricalProjection

class neuroml.nml.nml.**ElectricalProjection**(*id: a NmlId (required) = None*, *presynaptic_population: a NmlId (required) = None*, *postsynaptic_population: a NmlId (required) = None*, *electrical_connections: list of ElectricalConnection(s) (optional) = None*, *electrical_connection_instances: list of ElectricalConnectionInstance(s) (optional) = None*, *electrical_connection_instance_ws: list of ElectricalConnectionInstanceW(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseProjection*

ElectricalProjection – A projection between **presynapticPopulation** to another **postsynapticPopulation** through gap junctions.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

exportHdf5(*h5file*, *h5Group*)

Export to HDF5 file.

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ExpCondSynapse

class neuroml.nml.nml.**ExpCondSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *tau_syn: a float (required) = None*, *e_rev: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BasePynnSynapse*

ExpCondSynapse – Conductance based synapse with instantaneous rise and single exponential decay (with time constant tau_syn)

Parameters

- **e_rev** (*none*) –
- **tau_syn** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ExpCurrSynapse

class neuroml.nml.nml.**ExpCurrSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *tau_syn: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BasePynnSynapse*

ExpCurrSynapse – Current based synapse with instantaneous rise and single exponential decay (with time constant tau_syn)

Parameters
tau_syn (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ExpOneSynapse

class neuroml.nml.nml.**ExpOneSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *gbase: a Nml2Quantity_conductance (required) = None*, *erev: a Nml2Quantity_voltage (required) = None*, *tau_decay: a Nml2Quantity_time (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: BaseConductanceBasedSynapse

ExpOneSynapse – Ohmic synapse model whose conductance rises instantaneously by (**gbase** * **weight**) on receiving an event, and which decays exponentially to zero with time course **tauDecay**

Parameters

- **tauDecay** (*time*) – Time course of decay
- **gbase** (*conductance*) – Baseline conductance, generally the maximum conductance following a single spike
- **erev** (*voltage*) – Reversal potential of the synapse

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ExpThreeSynapse

class neuroml.nml.nml.**ExpThreeSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *gbase1: a Nml2Quantity_conductance (required) = None*, *gbase2: a Nml2Quantity_conductance (required) = None*, *erev: a Nml2Quantity_voltage (required) = None*, *tau_decay1: a Nml2Quantity_time (required) = None*, *tau_decay2: a Nml2Quantity_time (required) = None*, *tau_rise: a Nml2Quantity_time (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseConductanceBasedSynapseTwo*

ExpThreeSynapse – Ohmic synapse similar to expTwoSynapse but consisting of two components that can differ in decay times and max conductances but share the same rise time.

Parameters

- **tauRise** (*time*) –
- **tauDecay1** (*time*) –
- **tauDecay2** (*time*) –
- **gbase1** (*conductance*) – Baseline conductance 1
- **gbase2** (*conductance*) – Baseline conductance 2
- **erev** (*voltage*) – Reversal potential of the synapse

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ExpTwoSynapse

class neuroml.nml.nml.**ExpTwoSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *gbase: a Nml2Quantity_conductance (required) = None*, *erev: a Nml2Quantity_voltage (required) = None*, *tau_decay: a Nml2Quantity_time (required) = None*, *tau_rise: a Nml2Quantity_time (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseConductanceBasedSynapse*

ExpTwoSynapse – Ohmic synapse model whose conductance waveform on receiving an event has a rise time of **tauRise** and a decay time of **tauDecay.** Max conductance reached during this time (assuming zero conductance before) is **gbase** * **weight.**

Parameters

- **tauRise** (*time*) –
- **tauDecay** (*time*) –
- **gbase** (*conductance*) – Baseline conductance, generally the maximum conductance following a single spike
- **erev** (*voltage*) – Reversal potential of the synapse

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ExplicitInput

class neuroml.nml.nml.**ExplicitInput**(*target: a string (required) = None*, *input: a string (required) = None*, *destination: a string (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

ExplicitInput – An explicit input (anything which extends **basePointCurrent**) to a target cell in a population

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

get_fraction_along()

Get fraction along.

Returns 0.5 is fraction_along was not set.

get_segment_id()

Get the ID of the segment.

Returns 0 if segment_id was not set.

get_target_cell_id()

Get target cell ID

get_target_population()

Get target population.

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Exposure

class neuroml.nml.nml.**Exposure**(*name: a string (required) = None*, *dimension: a string (required) = None*, *description: a string (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Exposure – LEMS Exposure (ComponentType property)

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ExtracellularProperties

class neuroml.nml.nml.**ExtracellularProperties**(*id: a NmlId (required) = None*, *species: list of Species(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an

information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ExtracellularPropertiesLocal

`class` neuroml.nml.nml.`ExtracellularPropertiesLocal`(*id: a NmlId (required) = None*, *species: list of Species(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

FitzHughNagumo1969Cell

class neuroml.nml.nml.**FitzHughNagumo1969Cell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *a: a Nml2Quantity_none (required) = None*, *b: a Nml2Quantity_none (required) = None*, *I: a Nml2Quantity_none (required) = None*, *phi: a Nml2Quantity_none (required) = None*, *V0: a Nml2Quantity_none (required) = None*, *W0: a Nml2Quantity_none (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseCell*

FitzHughNagumo1969Cell – The Fitzhugh Nagumo model is a two-dimensional simplification of the Hodgkin-Huxley model of spike generation in squid giant axons. This system was suggested by FitzHugh (FitzHugh R. [1961]: Impulses and physiological states in theoretical models of nerve membrane. Biophysical J. 1:445-466), who called it " Bonhoeffer-van der Pol model ", and the equivalent circuit by Nagumo et al. (Nagumo J. , Arimoto S. , and Yoshizawa S. [1962] An active pulse transmission line simulating nerve axon. Proc IRE. 50:2061-2070. 1962). This version corresponds to the one described in FitzHugh R. [1969]: Mathematical models of excitation and propagation in nerve. Chapter 1 (pp. 1-85 in H. P. Schwan, ed. Biological Engineering, McGraw-Hill Book Co. , N. Y.)

Parameters

- **a** (*none*) –
- **b** (*none*) –
- **I** (*none*) – plays the role of an external injected current
- **phi** (*none*) –
- **V0** (*none*) –
- **W0** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters

return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

FitzHughNagumoCell

`class` neuroml.nml.nml.`FitzHughNagumoCell`(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *I: a Nml2Quantity_none (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseCell*

FitzHughNagumoCell – Simple dimensionless model of spiking cell from FitzHugh and Nagumo. Superseded by **fitzHughNagumo1969Cell** (See https://github.com/NeuroML/NeuroML2/issues/42)

Parameters
I (*none*) –

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to

- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

FixedFactorConcentrationModel

class neuroml.nml.nml.**FixedFactorConcentrationModel**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *ion: a NmlId (required) = None*, *resting_conc: a Nml2Quantity_concentration (required) = None*, *decay_constant: a Nml2Quantity_time (required) = None*, *rho: a Nml2Quantity_rhoFactor (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

FixedFactorConcentrationModel – Model of buffering of concentration of an ion (currently hard coded to be calcium, due to requirement for **iCa**) which has a baseline level **restingConc** and tends to this value with time course **decayConstant.** A fixed factor **rho** is used to scale the incoming current *independently of the size of the compartment* to produce a concentration change.

Parameters

- **restingConc** (*concentration*) –
- **decayConstant** (*time*) –
- **rho** (*rho_factor*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None

- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ForwardTransition

class neuroml.nml.nml.**ForwardTransition**(*id: a NmlId (required) = None*, *from_: a NmlId (required) = None*, *to: a NmlId (required) = None*, *anytypeobjs_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

ForwardTransition – A forward only **KSTransition** for a **gateKS** which specifies a **rate** (type **baseHHRate**) which follows one of the standard Hodgkin Huxley forms (e. g. **HHExpRate** , **HHSigmoidRate** , **HHExpLinearRate**

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found

- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GapJunction

class neuroml.nml.nml.**GapJunction**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *conductance: a Nml2Quantity_conductance (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseSynapse*

GapJunction – Gap junction/single electrical connection

Parameters
conductance (*conductance*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GateFractional

class neuroml.nml.nml.**GateFractional**(*id: a NmlId (required) = None*, *instances: a PositiveInteger (required) = None*, *notes: a string (optional) = None*, *q10_settings: a Q10Settings (optional) = None*, *sub_gates: list of GateFractionalSubgate(s) (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

GateFractional – Gate composed of subgates contributing with fractional conductance

Parameters
instances (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GateFractionalSubgate

class neuroml.nml.nml.**GateFractionalSubgate**(*id: a NmlId (required) = None*, *fractional_conductance: a Nml2Quantity_none (required) = None*, *notes: a string (optional) = None*, *q10_settings: a Q10Settings (optional) = None*, *steady_state: a HHVariable (required) = None*, *time_course: a HHTime (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GateHHInstantaneous

class neuroml.nml.nml.**GateHHInstantaneous**(*id: a NmlId (required) = None*, *instances: a PositiveInteger (required) = None*, *notes: a string (optional) = None*, *steady_state: a HHVariable (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

GateHHInstantaneous – Gate which follows the general Hodgkin Huxley formalism but is instantaneous, so tau = 0 and gate follows exactly inf value

Parameters
instances (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child

elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GateHHRates

class neuroml.nml.nml.**GateHHRates**(*id: a NmlId (required) = None*, *instances: a PositiveInteger (required) = None*, *notes: a string (optional) = None*, *q10_settings: a Q10Settings (optional) = None*, *forward_rate: a HHRate (required) = None*, *reverse_rate: a HHRate (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

GateHHRates – Gate which follows the general Hodgkin Huxley formalism

Parameters
instances (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GateHHRatesInf

class neuroml.nml.nml.**GateHHRatesInf**(*id: a NmlId (required) = None*, *instances: a PositiveInteger (required) = None*, *notes: a string (optional) = None*, *q10_settings: a Q10Settings (optional) = None*, *forward_rate: a HHRate (required) = None*, *reverse_rate: a HHRate (required) = None*, *steady_state: a HHVariable (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

GateHHRatesInf – Gate which follows the general Hodgkin Huxley formalism

Parameters
instances (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GateHHRatesTau

class neuroml.nml.nml.**GateHHRatesTau**(*id: a NmlId (required) = None*, *instances: a PositiveInteger (required) = None*, *notes: a string (optional) = None*, *q10_settings: a Q10Settings (optional) = None*, *forward_rate: a HHRate (required) = None*, *reverse_rate: a HHRate (required) = None*, *time_course: a HHTime (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

GateHHRatesTau – Gate which follows the general Hodgkin Huxley formalism

Parameters
instances (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GateHHRatesTauInf

class neuroml.nml.nml.**GateHHRatesTauInf**(*id: a NmlId (required) = None*, *instances: a PositiveInteger (required) = None*, *notes: a string (optional) = None*, *q10_settings: a Q10Settings (optional) = None*, *forward_rate: a HHRate (required) = None*, *reverse_rate: a HHRate (required) = None*, *time_course: a HHTime (required) = None*, *steady_state: a HHVariable (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: Base

GateHHRatesTauInf – Gate which follows the general Hodgkin Huxley formalism

Parameters
instances (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GateHHTauInf

class neuroml.nml.nml.**GateHHTauInf**(*id: a NmlId (required) = None*, *instances: a PositiveInteger (required) = None*, *notes: a string (optional) = None*, *q10_settings: a Q10Settings (optional) = None*, *time_course: a HHTime (required) = None*, *steady_state: a HHVariable (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

GateHHTauInf – Gate which follows the general Hodgkin Huxley formalism

Parameters
instances (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters

return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GateHHUndetermined

`class` neuroml.nml.nml.**GateHHUndetermined**(*id: a NmlId (required) = None*, *instances: a PositiveInteger (required) = None*, *type: a gateTypes (required) = None*, *notes: a string (optional) = None*, *q10_settings: a Q10Settings (optional) = None*, *forward_rate: a HHRate (optional) = None*, *reverse_rate: a HHRate (optional) = None*, *time_course: a HHTime (optional) = None*, *steady_state: a HHVariable (optional) = None*, *sub_gates: list of GateFractionalSubgate(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

GateHHUndetermined – Note all sub elements for gateHHrates, gateHHratesTau, gateFractional etc. allowed here. Which are valid should be constrained by what type is set

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to

- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GateKS

class neuroml.nml.nml.**GateKS**(*id: a NmlId (required) = None*, *instances: a PositiveInteger (required) = None*, *notes: a string (optional) = None*, *q10_settings: a Q10Settings (optional) = None*, *closed_states: list of ClosedState(s) (required) = None*, *open_states: list of OpenState(s) (required) = None*, *forward_transition: list of ForwardTransition(s) (required) = None*, *reverse_transition: list of ReverseTransition(s) (required) = None*, *tau_inf_transition: list of TauInfTransition(s) (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

GateKS – A gate which consists of multiple **KSState** s and **KSTransition** s giving the rates of transition between them

Parameters
instances (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GeneratedsSuper

class neuroml.nml.nml.**GeneratedsSuper**

Bases: *GeneratedsSuperSuper*

GradedSynapse

class neuroml.nml.nml.**GradedSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *conductance: a Nml2Quantity_conductance (required) = None*, *delta: a Nml2Quantity_voltage (required) = None*, *Vth: a Nml2Quantity_voltage (required) = None*, *k: a Nml2Quantity_pertime (required) = None*, *erev: a Nml2Quantity_voltage (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseSynapse*

GradedSynapse – Graded/analog synapse. Based on synapse in Methods of http://www.nature.com/neuro/journal/v7/n12/abs/nn1352.html

Parameters

- **conductance** (*conductance*) –
- **delta** (*voltage*) – Slope of the activation curve
- **k** (*per_time*) – Rate constant for transmitter-receptor dissociation rate
- **Vth** (*voltage*) – The half-activation voltage of the synapse
- **erev** (*voltage*) – The reversal potential of the synapse

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to

- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

GridLayout

class neuroml.nml.nml.**GridLayout**(*x_size: a nonNegativeInteger (optional) = None*, *y_size: a nonNegativeInteger (optional) = None*, *z_size: a nonNegativeInteger (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided,

only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

HHRate

class neuroml.nml.nml.**HHRate**(*type: a NmlId (required) = None*, *rate: a Nml2Quantity_pertime (optional) = None*, *midpoint: a Nml2Quantity_voltage (optional) = None*, *scale: a Nml2Quantity_voltage (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

HHTime

class neuroml.nml.nml.**HHTime**(*type: a NmlId (required) = None*, *rate: a Nml2Quantity_time (optional) = None*, *midpoint: a Nml2Quantity_voltage (optional) = None*, *scale: a Nml2Quantity_voltage (optional) = None*, *tau: a Nml2Quantity_time (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child

elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

HHVariable

class neuroml.nml.nml.**HHVariable**(*type: a NmlId (required) = None*, *rate: a float (optional) = None*, *midpoint: a Nml2Quantity_voltage (optional) = None*, *scale: a Nml2Quantity_voltage (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

HH_cond_exp

class neuroml.nml.nml.**HH_cond_exp**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *cm: a float (required) = None*, *i_offset: a float (required) = None*, *tau_syn_E: a float (required) = None*, *tau_syn_I: a float (required) = None*, *v_init: a float (required) = None*, *v_offset: a float (required) = None*, *e_rev_E: a float (required) = None*, *e_rev_I: a float (required) = None*, *e_rev_K: a float (required) = None*, *e_rev_Na: a float (required) = None*, *e_rev_leak: a float (required) = None*, *g_leak: a float (required) = None*, *gbar_K: a float (required) = None*, *gbar_Na: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *basePyNNCell*

HH_cond_exp – Single-compartment Hodgkin-Huxley-type neuron with transient sodium and delayed-rectifier potassium currents using the ion channel models from Traub.

Parameters

- **gbar_K** (*none*) –
- **gbar_Na** (*none*) –
- **g_leak** (*none*) –
- **e_rev_K** (*none*) –
- **e_rev_Na** (*none*) –
- **e_rev_leak** (*none*) –
- **v_offset** (*none*) –
- **e_rev_E** (*none*) –
- **e_rev_I** (*none*) –
- **cm** (*none*) –
- **i_offset** (*none*) –
- **tau_syn_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_syn_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **v_init** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters

return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IF_cond_alpha

`class` neuroml.nml.nml.`IF_cond_alpha`(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *cm: a float (required) = None*, *i_offset: a float (required) = None*, *tau_syn_E: a float (required) = None*, *tau_syn_I: a float (required) = None*, *v_init: a float (required) = None*, *tau_m: a float (required) = None*, *tau_refrac: a float (required) = None*, *v_reset: a float (required) = None*, *v_rest: a float (required) = None*, *v_thresh: a float (required) = None*, *e_rev_E: a float (required) = None*, *e_rev_I: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *basePyNNIaFCondCell*

IF_cond_alpha – Leaky integrate and fire model with fixed threshold and alpha-function-shaped post-synaptic conductance

Parameters

- **e_rev_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **e_rev_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_refrac** (*none*) –
- **v_thresh** (*none*) –
- **tau_m** (*none*) –
- **v_rest** (*none*) –

- **v_reset** (*none*) –
- **cm** (*none*) –
- **i_offset** (*none*) –
- **tau_syn_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_syn_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **v_init** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IF_cond_exp

class neuroml.nml.nml.**IF_cond_exp**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *cm: a float (required) = None*, *i_offset: a float (required) = None*, *tau_syn_E: a float (required) = None*, *tau_syn_I: a float (required) = None*, *v_init: a float (required) = None*, *tau_m: a float (required) = None*, *tau_refrac: a float (required) = None*, *v_reset: a float (required) = None*, *v_rest: a float (required) = None*, *v_thresh: a float (required) = None*, *e_rev_E: a float (required) = None*, *e_rev_I: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *basePyNNIaFCondCell*

IF_cond_exp – Leaky integrate and fire model with fixed threshold and exponentially-decaying post-synaptic conductance

Parameters

- **e_rev_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **e_rev_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_refrac** (*none*) –
- **v_thresh** (*none*) –
- **tau_m** (*none*) –
- **v_rest** (*none*) –
- **v_reset** (*none*) –
- **cm** (*none*) –
- **i_offset** (*none*) –
- **tau_syn_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_syn_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **v_init** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to

- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IF_curr_alpha

class neuroml.nml.nml.**IF_curr_alpha**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *cm: a float (required) = None*, *i_offset: a float (required) = None*, *tau_syn_E: a float (required) = None*, *tau_syn_I: a float (required) = None*, *v_init: a float (required) = None*, *tau_m: a float (required) = None*, *tau_refrac: a float (required) = None*, *v_reset: a float (required) = None*, *v_rest: a float (required) = None*, *v_thresh: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *basePyNNIaFCell*

IF_curr_alpha – Leaky integrate and fire model with fixed threshold and alpha-function-shaped post-synaptic current

Parameters

- **tau_refrac** (*none*) –
- **v_thresh** (*none*) –
- **tau_m** (*none*) –
- **v_rest** (*none*) –
- **v_reset** (*none*) –
- **cm** (*none*) –
- **i_offset** (*none*) –
- **tau_syn_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_syn_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **v_init** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IF_curr_exp

class neuroml.nml.nml.**IF_curr_exp**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *cm: a float (required) = None*, *i_offset: a float (required) = None*, *tau_syn_E: a float (required) = None*, *tau_syn_I: a float (required) = None*, *v_init: a float (required) = None*, *tau_m: a float (required) = None*, *tau_refrac: a float (required) = None*, *v_reset: a float (required) = None*, *v_rest: a float (required) = None*, *v_thresh: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *basePyNNIaFCell*

IF_curr_exp – Leaky integrate and fire model with fixed threshold and decaying-exponential post-synaptic current

Parameters

- **tau_refrac** (*none*) –
- **v_thresh** (*none*) –
- **tau_m** (*none*) –
- **v_rest** (*none*) –

- **v_reset** (*none*) –
- **cm** (*none*) –
- **i_offset** (*none*) –
- **tau_syn_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_syn_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **v_init** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IafCell

class neuroml.nml.nml.**IafCell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *leak_reversal: a Nml2Quantity_voltage (required) = None*, *thresh: a Nml2Quantity_voltage (required) = None*, *reset: a Nml2Quantity_voltage (required) = None*, *C: a Nml2Quantity_capacitance (required) = None*, *leak_conductance: a Nml2Quantity_conductance (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseCell*

IafCell – Integrate and fire cell with capacitance **C, leakConductance** and **leakReversal**

Parameters

- **leakConductance** (*conductance*) –
- **leakReversal** (*voltage*) –
- **thresh** (*voltage*) –
- **reset** (*voltage*) –
- **C** (*capacitance*) – Total capacitance of the cell membrane

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IafRefCell

class neuroml.nml.nml.**IafRefCell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *leak_reversal: a Nml2Quantity_voltage (required) = None*, *thresh: a Nml2Quantity_voltage (required) = None*, *reset: a Nml2Quantity_voltage (required) = None*, *C: a Nml2Quantity_capacitance (required) = None*, *leak_conductance: a Nml2Quantity_conductance (required) = None*, *refract: a Nml2Quantity_time (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *IafCell*

IafRefCell – Integrate and fire cell with capacitance **C, leakConductance, leakReversal** and refractory period **refract**

Parameters

- **refract** (*time*) –
- **leakConductance** (*conductance*) –
- **leakReversal** (*voltage*) –
- **thresh** (*voltage*) –
- **reset** (*voltage*) –
- **C** (*capacitance*) – Total capacitance of the cell membrane

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IafTauCell

class neuroml.nml.nml.**IafTauCell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *leak_reversal: a Nml2Quantity_voltage (required) = None*, *thresh: a Nml2Quantity_voltage (required) = None*, *reset: a Nml2Quantity_voltage (required) = None*, *tau: a Nml2Quantity_time (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseCell*

IafTauCell – Integrate and fire cell which returns to its leak reversal potential of **leakReversal** with a time constant **tau**

Parameters

- **leakReversal** (*voltage*) –
- **tau** (*time*) –
- **thresh** (*voltage*) – The membrane potential at which to emit a spiking event and reset voltage
- **reset** (*voltage*) – The value the membrane potential is reset to on spiking

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IafTauRefCell

class neuroml.nml.nml.**IafTauRefCell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *leak_reversal: a Nml2Quantity_voltage (required) = None*, *thresh: a Nml2Quantity_voltage (required) = None*, *reset: a Nml2Quantity_voltage (required) = None*, *tau: a Nml2Quantity_time (required) = None*, *refract: a Nml2Quantity_time (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *IafTauCell*

IafTauRefCell – Integrate and fire cell which returns to its leak reversal potential of **leakReversal** with a time course **tau.** It has a refractory period of **refract** after spiking

Parameters

- **refract** (*time*) –
- **leakReversal** (*voltage*) –
- **tau** (*time*) –
- **thresh** (*voltage*) – The membrane potential at which to emit a spiking event and reset voltage
- **reset** (*voltage*) – The value the membrane potential is reset to on spiking

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Include

class neuroml.nml.nml.**Include**(*segment_groups: a NmlId (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Include – Include all members of another **segmentGroup** in this group

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IncludeType

class neuroml.nml.nml.**IncludeType**(*href: a anyURI (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *GeneratedsSuper*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters

return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

InhomogeneousParameter

`class` neuroml.nml.nml.**InhomogeneousParameter**(*id: a NmlId (required) = None*, *variable: a string (required) = None*, *metric: a Metric (required) = None*, *proximal: a ProximalDetails (optional) = None*, *distal: a DistalDetails (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

InhomogeneousParameter – An inhomogeneous parameter specified across the **segmentGroup** (see **variableParameter** for usage).

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

InhomogeneousValue

class neuroml.nml.nml.**InhomogeneousValue**(*inhomogeneous_parameters: a string (required) = None*, *value: a string (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

InhomogeneousValue – Specifies the **value** of an **inhomogeneousParameter.** For usage see **variableParameter**

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)

- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

InitMembPotential

class neuroml.nml.nml.**InitMembPotential**(*value: a Nml2Quantity_voltage (required) = None*, *segment_groups: a NmlId (optional) = 'all'*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

InitMembPotential – Explicitly set initial membrane potential for the cell

Parameters
value (*voltage*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Input

class neuroml.nml.nml.**Input**(*id: a NonNegativeInteger (required) = None*, *target: a string (required) = None*, *destination: a NmlId (required) = None*, *segment_id: a NonNegativeInteger (optional) = None*, *fraction_along: a ZeroToOne (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseNonNegativeIntegerId*

Input – Specifies a single input to a **target,** optionally giving the **segmentId** (default 0) and **fractionAlong** the segment (default 0. 5).

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

get_fraction_along()

Get fraction along.

Returns 0.5 is fraction_along was not set.

get_segment_id()

Get the ID of the segment.

Returns 0 if segment_id was not set.

get_target_cell_id()

Get ID of target cell.

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

InputList

class neuroml.nml.nml.**InputList**(*id: a NonNegativeInteger (required) = None*, *populations: a NmlId (required) = None*, *component: a NmlId (required) = None*, *input: list of Input(s) (optional) = None*, *input_ws: list of InputW(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

InputList – An explicit list of **input** s to a **population.**

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

exportHdf5(*h5file*, *h5Group*)

Export to HDF5 file.

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids

can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

InputW

class neuroml.nml.nml.**InputW**(*id: a NonNegativeInteger (required) = None*, *target: a string (required) = None*, *destination: a NmlId (required) = None*, *segment_id: a NonNegativeInteger (optional) = None*, *fraction_along: a ZeroToOne (optional) = None*, *weight: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Input*

InputW – Specifies input lists. Can set **weight** to scale individual inputs.

Parameters
weight (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

get_fraction_along()

Get fraction along.

Returns 0.5 is fraction_along was not set.

get_segment_id()

Get the ID of the segment.

Returns 0 if segment_id was not set.

get_target_cell_id()

Get ID of target cell.

get_weight()

Get weight.

If weight is not set, the default value of 1.0 is returned.

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Instance

class neuroml.nml.nml.**Instance**(*id: a nonNegativeInteger (optional) = None*, *i: a nonNegativeInteger (optional) = None*, *j: a nonNegativeInteger (optional) = None*, *k: a nonNegativeInteger (optional) = None*, *location: a Location (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Instance – Specifies a single instance of a component in a **population** (placed at **location**).

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters

return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

InstanceRequirement

`class` neuroml.nml.nml.`InstanceRequirement`(*name: a string (required) = None*, *type: a string (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *GeneratedsSuper*

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found

- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IntracellularProperties

class neuroml.nml.nml.**IntracellularProperties**(*species: list of Species(s) (optional) = None*, *resistivities: list of Resistivity(s) (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

IntracellularProperties – Biophysical properties related to the intracellular space within the **cell** , such as the **resistivity** and the list of ionic **species** present. **caConc** and **caConcExt** are explicitly exposed here to facilitate accessing these values from other Components, even though **caConcExt** is clearly not an intracellular property

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IntracellularProperties2CaPools

class neuroml.nml.nml.**IntracellularProperties2CaPools**(*species: list of Species(s) (optional) = None*, *resistivities: list of Resistivity(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *IntracellularProperties*

IntracellularProperties2CaPools – Variant of intracellularProperties with 2 independent Ca pools

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need

to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IonChannel

class neuroml.nml.nml.**IonChannel**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *q10_conductance_scalings: list of Q10ConductanceScaling(s) (optional) = None*, *species: a NmlId (optional) = None*, *type: a channelTypes (optional) = None*, *conductance: a Nml2Quantity_conductance (optional) = None*, *gates: list of GateHHUndetermined(s) (optional) = None*, *gate_hh_rates: list of GateHHRates(s) (optional) = None*, *gate_h_hrates_taus: list of GateHHRatesTau(s) (optional) = None*, *gate_hh_tau_infs: list of GateHHTauInf(s) (optional) = None*, *gate_h_hrates_infs: list of GateHHRatesInf(s) (optional) = None*, *gate_h_hrates_tau_infs: list of GateHHRatesTauInf(s) (optional) = None*, *gate_hh_instantaneouses: list of GateHHInstantaneous(s) (optional) = None*, *gate_fractionals: list of GateFractional(s) (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *IonChannelScalable*

IonChannel – Note **ionChannel** and **ionChannelHH** are currently functionally identical. This is needed since many existing examples use ionChannel, some use ionChannelHH. NeuroML v2beta4 should remove one of these, probably ionChannelHH.

Parameters
conductance (*conductance*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IonChannelHH

class neuroml.nml.nml.**IonChannelHH**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *q10_conductance_scalings: list of Q10ConductanceScaling(s) (optional) = None*, *species: a NmlId (optional) = None*, *type: a channelTypes (optional) = None*, *conductance: a Nml2Quantity_conductance (optional) = None*, *gates: list of GateHHUndetermined(s) (optional) = None*, *gate_hh_rates: list of GateHHRates(s) (optional) = None*, *gate_h_hrates_taus: list of GateHHRatesTau(s) (optional) = None*, *gate_hh_tau_infs: list of GateHHTauInf(s) (optional) = None*, *gate_h_hrates_infs: list of GateHHRatesInf(s) (optional) = None*, *gate_h_hrates_tau_infs: list of GateHHRatesTauInf(s) (optional) = None*, *gate_hh_instantaneouses: list of GateHHInstantaneous(s) (optional) = None*, *gate_fractionals: list of GateFractional(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *IonChannel*

IonChannelHH – Note **ionChannel** and **ionChannelHH** are currently functionally identical. This is needed since many existing examples use ionChannel, some use ionChannelHH. NeuroML v2beta4 should remove one of these, probably ionChannelHH.

Parameters
conductance (*conductance*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IonChannelKS

class neuroml.nml.nml.**IonChannelKS**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *species: a NmlId (optional) = None*, *conductance: a Nml2Quantity_conductance (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *gate_kses: list of GateKS(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

A kinetic scheme based ion channel with multiple **gateKS** s, each of which consists of multiple **KSState** s and **KSTransition** s giving the rates of transition between them IonChannelKS – A kinetic scheme based ion channel with multiple **gateKS** s, each of which consists of multiple **KSState** s and **KSTransition** s giving the rates of transition between them

Parameters
conductance (*conductance*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IonChannelScalable

class neuroml.nml.nml.**IonChannelScalable**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *q10_conductance_scalings: list of Q10ConductanceScaling(s) (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IonChannelVShift

class neuroml.nml.nml.**IonChannelVShift**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *q10_conductance_scalings: list of Q10ConductanceScaling(s) (optional) = None*, *species: a NmlId (optional) = None*, *type: a channelTypes (optional) = None*, *conductance: a Nml2Quantity_conductance (optional) = None*, *gates: list of GateHHUndetermined(s) (optional) = None*, *gate_hh_rates: list of GateHHRates(s) (optional) = None*, *gate_h_hrates_taus: list of GateHHRatesTau(s) (optional) = None*, *gate_hh_tau_infs: list of GateHHTauInf(s) (optional) = None*, *gate_h_hrates_infs: list of GateHHRatesInf(s) (optional) = None*, *gate_h_hrates_tau_infs: list of GateHHRatesTauInf(s) (optional) = None*, *gate_hh_instantaneouses: list of GateHHInstantaneous(s) (optional) = None*, *gate_fractionals: list of GateFractional(s) (optional) = None*, *v_shift: a Nml2Quantity_voltage (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *IonChannel*

IonChannelVShift – Same as **ionChannel** , but with a **vShift** parameter to change voltage activation of gates. The exact usage of **vShift** in expressions for rates is determined by the individual gates.

Parameters

- **vShift** (*voltage*) –
- **conductance** (*conductance*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found

- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Izhikevich2007Cell

class neuroml.nml.nml.**Izhikevich2007Cell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *C: a Nml2Quantity_capacitance (required) = None*, *v0: a Nml2Quantity_voltage (required) = None*, *k: a Nml2Quantity_conductancePerVoltage (required) = None*, *vr: a Nml2Quantity_voltage (required) = None*, *vt: a Nml2Quantity_voltage (required) = None*, *vpeak: a Nml2Quantity_voltage (required) = None*, *a: a Nml2Quantity_pertime (required) = None*, *b: a Nml2Quantity_conductance (required) = None*, *c: a Nml2Quantity_voltage (required) = None*, *d: a Nml2Quantity_current (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseCellMembPotCap*

Izhikevich2007Cell – Cell based on the modified Izhikevich model in Izhikevich 2007, Dynamical systems in neuroscience, MIT Press

Parameters

- **v0** (*voltage*) –
- **k** (*conductance_per_voltage*) –
- **vr** (*voltage*) –
- **vt** (*voltage*) –
- **vpeak** (*voltage*) –
- **a** (*per_time*) –
- **b** (*conductance*) –
- **c** (*voltage*) –
- **d** (*current*) –
- **C** (*capacitance*) – Total capacitance of the cell membrane

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child

elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

IzhikevichCell

class neuroml.nml.nml.**IzhikevichCell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *v0: a Nml2Quantity_voltage (required) = None*, *thresh: a Nml2Quantity_voltage (required) = None*, *a: a Nml2Quantity_none (required) = None*, *b: a Nml2Quantity_none (required) = None*, *c: a Nml2Quantity_none (required) = None*, *d: a Nml2Quantity_none (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseCell*

IzhikevichCell – Cell based on the 2003 model of Izhikevich, see http://izhikevich.org/publications/spikes.htm

Parameters

- **v0** (*voltage*) – Initial membrane potential
- **a** (*none*) – Time scale of the recovery variable U
- **b** (*none*) – Sensitivity of U to the subthreshold fluctuations of the membrane potential V
- **c** (*none*) – After-spike reset value of V
- **d** (*none*) – After-spike increase to U
- **thresh** (*voltage*) – Spike threshold

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child

elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

LEMS_Property

class neuroml.nml.nml.**LEMS_Property**(*name: a string (required) = None*, *dimension: a string (required) = None*, *description: a string (optional) = None*, *default_value: a double (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *NamedDimensionalType*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Layout

class neuroml.nml.nml.**Layout**(*spaces: a NmlId (optional) = None*, *random: a RandomLayout (required) = None*, *grid: a GridLayout (required) = None*, *unstructured: a UnstructuredLayout (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

LinearGradedSynapse

`class` neuroml.nml.nml.`LinearGradedSynapse`(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *conductance: a Nml2Quantity_conductance (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseSynapse*

LinearGradedSynapse – Behaves just like a one way gap junction.

Parameters
conductance (*conductance*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)

- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Location

class neuroml.nml.nml.**Location**(*x: a float (required) = None*, *y: a float (required) = None*, *z: a float (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Location – Specifies the (x, y, z) location of a single **instance** of a component in a **population**

Parameters

- **x** (*none*) –
- **y** (*none*) –
- **z** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters

return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Member

`class` neuroml.nml.nml.`Member`(*segments: a NonNegativeInteger (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Member – A single identified **segment** which is part of the **segmentGroup**

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

MembraneProperties

class neuroml.nml.nml.**MembraneProperties**(*channel_populations: list of ChannelPopulation(s) (optional) = None*, *channel_densities: list of ChannelDensity(s) (optional) = None*, *channel_density_v_shifts: list of ChannelDensityVShift(s) (optional) = None*, *channel_density_nernsts: list of ChannelDensityNernst(s) (optional) = None*, *channel_density_ghks: list of ChannelDensityGHK(s) (optional) = None*, *channel_density_ghk2s: list of ChannelDensityGHK2(s) (optional) = None*, *channel_density_non_uniforms: list of ChannelDensityNonUniform(s) (optional) = None*, *channel_density_non_uniform_nernsts: list of ChannelDensityNonUniformNernst(s) (optional) = None*, *channel_density_non_uniform_ghks: list of ChannelDensityNonUniformGHK(s) (optional) = None*, *spike_threshes: list of SpikeThresh(s) (optional) = None*, *specific_capacitances: list of SpecificCapacitance(s) (optional) = None*, *init_memb_potentials: list of InitMembPotential(s) (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

MembraneProperties – Properties specific to the membrane, such as the **populations** of channels, **channelDensities, specificCapacitance,** etc.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

MembraneProperties2CaPools

class neuroml.nml.nml.**MembraneProperties2CaPools**(*channel_populations: list of ChannelPopulation(s) (optional) = None*, *channel_densities: list of ChannelDensity(s) (optional) = None*, *channel_density_v_shifts: list of ChannelDensityVShift(s) (optional) = None*, *channel_density_nernsts: list of ChannelDensityNernst(s) (optional) = None*, *channel_density_ghks: list of ChannelDensityGHK(s) (optional) = None*, *channel_density_ghk2s: list of ChannelDensityGHK2(s) (optional) = None*, *channel_density_non_uniforms: list of ChannelDensityNonUniform(s) (optional) = None*, *channel_density_non_uniform_nernsts: list of ChannelDensityNonUniformNernst(s) (optional) = None*, *channel_density_non_uniform_ghks: list of ChannelDensityNonUniformGHK(s) (optional) = None*, *spike_threshes: list of SpikeThresh(s) (optional) = None*, *specific_capacitances: list of SpecificCapacitance(s) (optional) = None*, *init_memb_potentials: list of InitMembPotential(s) (optional) = None*, *channel_density_nernst_ca2s: list of ChannelDensityNernstCa2(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *MembraneProperties*

MembraneProperties2CaPools – Variant of membraneProperties with 2 independent Ca pools

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Morphology

class neuroml.nml.nml.**Morphology**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *segments: list of Segment(s) (required) = None*, *segment_groups: list of SegmentGroup(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

Morphology – The collection of **segment** s which specify the 3D structure of the cell, along with a number of **segmentGroup** s

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

property num_segments

Get the number of segments included in this cell morphology.

Returns
number of segments

Return type
int

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

NamedDimensionalType

class neuroml.nml.nml.**NamedDimensionalType**(*name: a string (required) = None*, *dimension: a string (required) = None*, *description: a string (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

NamedDimensionalVariable

`class` neuroml.nml.nml.**NamedDimensionalVariable**(*name: a string (required) = None*, *dimension: a string (required) = None*, *description: a string (optional) = None*, *exposure: a string (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Network

class neuroml.nml.nml.**Network**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *type: a networkTypes (optional) = None*, *temperature: a Nml2Quantity_temperature (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *spaces: list of Space(s) (optional) = None*, *regions: list of Region(s) (optional) = None*, *extracellular_properties: list of ExtracellularPropertiesLocal(s) (optional) = None*, *populations: list of Population(s) (required) = None*, *cell_sets: list of CellSet(s) (optional) = None*, *synaptic_connections: list of SynapticConnection(s) (optional) = None*, *projections: list of Projection(s) (optional) = None*, *electrical_projections: list of ElectricalProjection(s) (optional) = None*, *continuous_projections: list of ContinuousProjection(s) (optional) = None*, *explicit_inputs: list of ExplicitInput(s) (optional) = None*, *input_lists: list of InputList(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

Network – Network containing: **population** s (potentially of type **populationList** , and so specifying a list of cell **location** s); **projection** s (with lists of **connection** s) and/or **explicitConnection** s; and **inputList** s (with lists of **input** s) and/or **explicitInput** s. Note: often in NeuroML this will be of type **networkWithTemperature** if there are temperature dependent elements (e. g. ion channels).

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

exportHdf5(*h5file*, *h5Group*)

Export to HDF5 file.

get_by_id(*id*)

Get a component by its ID

Parameters
id (*str*) – ID of component to find

Returns
component with specified ID or None if no component with specified ID found

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

NeuroMLDocument

class neuroml.nml.nml.**NeuroMLDocument**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *includes: list of IncludeType(s) (optional) = None*, *extracellular_properties: list of ExtracellularProperties(s) (optional) = None*, *intracellular_properties: list of IntracellularProperties(s) (optional) = None*, *morphology: list of Morphology(s) (optional) = None*, *ion_channel: list of IonChannel(s) (optional) = None*, *ion_channel_hhs: list of IonChannelHH(s) (optional) = None*, *ion_channel_v_shifts: list of IonChannelVShift(s) (optional) = None*, *ion_channel_kses: list of IonChannelKS(s) (optional) = None*, *decaying_pool_concentration_models: list of DecayingPoolConcentrationModel(s) (optional) = None*, *fixed_factor_concentration_models: list of FixedFactorConcentrationModel(s) (optional) = None*, *alpha_current_synapses: list of AlphaCurrentSynapse(s) (optional) = None*, *alpha_synapses: list of AlphaSynapse(s) (optional) = None*, *exp_one_synapses: list of ExpOneSynapse(s) (optional) = None*, *exp_two_synapses: list of ExpTwoSynapse(s) (optional) = None*, *exp_three_synapses: list of ExpThreeSynapse(s) (optional) = None*, *blocking_plastic_synapses: list of BlockingPlasticSynapse(s) (optional) = None*, *double_synapses: list of DoubleSynapse(s) (optional) = None*, *gap_junctions: list of GapJunction(s) (optional) = None*, *silent_synapses: list of SilentSynapse(s) (optional) = None*, *linear_graded_synapses: list of LinearGradedSynapse(s) (optional) = None*, *graded_synapses: list of GradedSynapse(s) (optional) = None*, *biophysical_properties: list of BiophysicalProperties(s) (optional) = None*, *cells: list of Cell(s) (optional) = None*, *cell2_ca_poolses: list of Cell2CaPools(s) (optional) = None*, *base_cells: list of BaseCell(s) (optional) = None*, *iaf_tau_cells: list of IafTauCell(s) (optional) = None*, *iaf_tau_ref_cells: list of IafTauRefCell(s) (optional) = None*, *iaf_cells: list of IafCell(s) (optional) = None*, *iaf_ref_cells: list of IafRefCell(s) (optional) = None*, *izhikevich_cells: list of IzhikevichCell(s) (optional) = None*, *izhikevich2007_cells: list of Izhikevich2007Cell(s) (optional) = None*, *ad_ex_ia_f_cells: list of AdExIaFCell(s) (optional) = None*, *fitz_hugh_nagumo_cells: list of FitzHughNagumoCell(s) (optional) = None*, *fitz_hugh_nagumo1969_cells: list of FitzHughNagumo1969Cell(s) (optional) = None*, *pinsky_rinzel_ca3_cells: list of PinskyRinzelCA3Cell(s) (optional) = None*, *pulse_generators: list of PulseGenerator(s) (optional) = None*, *pulse_generator_dls: list of PulseGeneratorDL(s) (optional) = None*, *sine_generators: list of SineGenerator(s) (optional) = None*, *sine_generator_dls: list of SineGeneratorDL(s) (optional) = None*, *ramp_generators: list of RampGenerator(s) (optional) = None*, *ramp_generator_dls: list of RampGeneratorDL(s) (optional) = None*, *compound_inputs: list of CompoundInput(s) (optional) = None*, *compound_input_dls: list of CompoundInputDL(s) (optional) = None*, *voltage_clamps: list of VoltageClamp(s) (optional) = None*, *voltage_clamp_triples: list of VoltageClampTriple(s) (optional) = None*, *spike_arrays: list of SpikeArray(s) (optional) = None*, *timed_synaptic_inputs: list of TimedSynapticInput(s) (optional) = None*, *spike_generators: list of SpikeGenerator(s) (optional) = None*, *spike_generator_randoms: list of SpikeGeneratorRandom(s) (optional) = None*, *spike_generator_poissons: list of SpikeGeneratorPoisson(s) (optional) = None*, *spike_generator_ref_poissons: list of SpikeGeneratorRefPoisson(s) (optional) = None*, *poisson_firing_synapses: list of PoissonFiringSynapse(s)*

Bases: *Standalone*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

append(*element*)

Append an element

Parameters
element (*Object*) – element to append

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need

to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

get_by_id(*id*)

Get a component by specifying its ID.

Parameters
id (*str*) – id of Component to get

Returns
Component with given ID or None if no Component with provided ID was found

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

summary(*show_includes=True*, *show_non_network=True*)

Get a pretty-printed summary of the complete NeuroMLDocument.

This includes information on the various Components included in the NeuroMLDocument: networks, cells, projections, synapses, and so on.

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

OpenState

class neuroml.nml.nml.**OpenState**(*id: a NmlId (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

OpenState – A **KSState** with **relativeConductance** of 1

Parameters
relativeConductance (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need

to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Parameter

class neuroml.nml.nml.**Parameter**(*name: a string (required) = None*, *dimension: a string (required) = None*, *description: a string (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *NamedDimensionalType*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters

return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Path

`class` neuroml.nml.nml.`Path`(*from_: a SegmentEndPoint (optional) = None*, *to: a SegmentEndPoint (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Path – Include all the **segment** s between those specified by **from** and **to** , inclusive

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

PinskyRinzelCA3Cell

class neuroml.nml.nml.**PinskyRinzelCA3Cell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *i_soma: a Nml2Quantity_currentDensity (required) = None*, *i_dend: a Nml2Quantity_currentDensity (required) = None*, *gc: a Nml2Quantity_conductanceDensity (required) = None*, *g_ls: a Nml2Quantity_conductanceDensity (required) = None*, *g_ld: a Nml2Quantity_conductanceDensity (required) = None*, *g_na: a Nml2Quantity_conductanceDensity (required) = None*, *g_kdr: a Nml2Quantity_conductanceDensity (required) = None*, *g_ca: a Nml2Quantity_conductanceDensity (required) = None*, *g_kahp: a Nml2Quantity_conductanceDensity (required) = None*, *g_kc: a Nml2Quantity_conductanceDensity (required) = None*, *g_nmda: a Nml2Quantity_conductanceDensity (required) = None*, *g_ampa: a Nml2Quantity_conductanceDensity (required) = None*, *e_na: a Nml2Quantity_voltage (required) = None*, *e_ca: a Nml2Quantity_voltage (required) = None*, *e_k: a Nml2Quantity_voltage (required) = None*, *e_l: a Nml2Quantity_voltage (required) = None*, *qd0: a Nml2Quantity_none (required) = None*, *pp: a Nml2Quantity_none (required) = None*, *alphac: a Nml2Quantity_none (required) = None*, *betac: a Nml2Quantity_none (required) = None*, *cm: a Nml2Quantity_specificCapacitance (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseCell*

PinskyRinzelCA3Cell – Reduced CA3 cell model from Pinsky and Rinzel 1994. See https://github.com/OpenSourceBrain/PinskyRinzelModel

Parameters

- **iSoma** (*currentDensity*) –
- **iDend** (*currentDensity*) –
- **gLs** (*conductanceDensity*) –
- **gLd** (*conductanceDensity*) –
- **gNa** (*conductanceDensity*) –
- **gKdr** (*conductanceDensity*) –
- **gCa** (*conductanceDensity*) –
- **gKahp** (*conductanceDensity*) –

- **gKC** (*conductanceDensity*) –
- **gc** (*conductanceDensity*) –
- **eNa** (*voltage*) –
- **eCa** (*voltage*) –
- **eK** (*voltage*) –
- **eL** (*voltage*) –
- **pp** (*none*) –
- **cm** (*specificCapacitance*) –
- **alphac** (*none*) –
- **betac** (*none*) –
- **gNmda** (*conductanceDensity*) –
- **gAmpa** (*conductanceDensity*) –
- **qd0** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)

- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

PlasticityMechanism

class neuroml.nml.nml.**PlasticityMechanism**(*type: a PlasticityTypes (required) = None*, *init_release_prob: a ZeroToOne (required) = None*, *tau_rec: a Nml2Quantity_time (required) = None*, *tau_fac: a Nml2Quantity_time (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need

to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Point3DWithDiam

class neuroml.nml.nml.**Point3DWithDiam**(*x: a double (required) = None*, *y: a double (required) = None*, *z: a double (required) = None*, *diameter: a DoubleGreaterThanZero (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Point3DWithDiam – Base type for ComponentTypes which specify an (**x, y, z**) coordinate along with a **diameter.** Note: no dimension used in the attributes for these coordinates! These are assumed to have dimension micrometer (10^-6 m). This is due to micrometers being the default option for the majority of neuronal morphology formats, and dimensions are omitted here to facilitate reading and writing of morphologies in NeuroML.

Parameters

- **x** (*none*) – x coordinate of the point. Note: no dimension used, see description of **point3DWithDiam** for details.

- **y** (*none*) – y coordinate of the ppoint. Note: no dimension used, see description of **point3DWithDiam** for details.
- **z** (*none*) – z coordinate of the ppoint. Note: no dimension used, see description of **point3DWithDiam** for details.
- **diameter** (*none*) – Diameter of the ppoint. Note: no dimension used, see description of **point3DWithDiam** for details.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

distance_to(*other_3d_point*)

Find the distance between this point and another.

Parameters
other_3d_point (Point3DWithDiam) – other 3D point to calculate distance to

Returns
distance between the two points

Return type
float

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

PoissonFiringSynapse

class neuroml.nml.nml.**PoissonFiringSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *average_rate: a Nml2Quantity_pertime (required) = None*, *synapse: a string (required) = None*, *spike_target: a string (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

PoissonFiringSynapse – Poisson spike generator firing at **averageRate,** which is connected to single **synapse** that is triggered every time a spike is generated, producing an input current. See also **transientPoissonFiringSynapse** .

Parameters

averageRate (*per_time*) – The average rate at which spikes are emitted

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Population

class neuroml.nml.nml.**Population**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *component: a NmlId (required) = None*, *size: a NonNegativeInteger (optional) = None*, *type: a populationTypes (optional) = None*, *extracellular_properties: a NmlId (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *layout: a Layout (optional) = None*, *instances: list of Instance(s) (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

Population – A population of components, with just one parameter for the **size,** i. e. number of components to create. Note: quite often this is used with type= **populationList** which means the size is determined by the number of **instance** s (with **location** s) in the list. The **size** attribute is still set, and there will be a validation error if this does not match the number in the list.

Parameters
size (*none*) – Number of instances of this Component to create when the population is instantiated

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

exportHdf5(*h5file*, *h5Group*)

Export to HDF5 file.

get_size()

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be

returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Projection

class neuroml.nml.nml.**Projection**(*id: a NmlId (required) = None*, *presynaptic_population: a NmlId (required) = None*, *postsynaptic_population: a NmlId (required) = None*, *synapse: a NmlId (required) = None*, *connections: list of Connection(s) (optional) = None*, *connection_wds: list of ConnectionWD(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseProjection*

Projection – Projection from one population, **presynapticPopulation** to another, **postsynapticPopulation,** through **synapse.** Contains lists of **connection** or **connectionWD** elements.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

exportHdf5(*h5file*, *h5Group*)

Export to HDF5 file.

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Property

class neuroml.nml.nml.**Property**(*tag: a string (required) = None*, *value: a string (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Property – A property (a **tag** and **value** pair), which can be on any **baseStandalone** either as a direct child, or within an **Annotation** . Generally something which helps the visual display or facilitates simulation of a Component, but is not a core physiological property. Common examples include: **numberInternalDivisions,** equivalent of nseg in NEURON; **radius,** for a radius to use in graphical displays for abstract cells (i. e. without defined morphologies); **color,** the color to use for a **Population** or **populationList** of cells; **recommended_dt_ms,** the recommended timestep to use for simulating a **Network** , **recommended_duration_ms** the recommended duration to use when running a **Network**

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ProximalDetails

class neuroml.nml.nml.**ProximalDetails**(*translation_start: a double (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an

information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

PulseGenerator

`class` neuroml.nml.nml.**PulseGenerator**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *delay: a Nml2Quantity_time (required) = None*, *duration: a Nml2Quantity_time (required) = None*, *amplitude: a Nml2Quantity_current (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

PulseGenerator – Generates a constant current pulse of a certain **amplitude** for a specified **duration** after a **delay.** Scaled by **weight,** if set

Parameters

- **delay** (*time*) – Delay before change in current. Current is zero prior to this.
- **duration** (*time*) – Duration for holding current at amplitude. Current is zero after delay + duration.
- **amplitude** (*current*) – Amplitude of current pulse

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters

return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

PulseGeneratorDL

`class` neuroml.nml.nml.**PulseGeneratorDL**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *delay: a Nml2Quantity_time (required) = None*, *duration: a Nml2Quantity_time (required) = None*, *amplitude: a Nml2Quantity_current (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

PulseGeneratorDL – Dimensionless equivalent of **pulseGenerator** . Generates a constant current pulse of a certain **amplitude** for a specified **duration** after a **delay.** Scaled by **weight,** if set

Parameters

- **delay** (*time*) – Delay before change in current. Current is zero prior to this.
- **duration** (*time*) – Duration for holding current at amplitude. Current is zero after delay + duration.
- **amplitude** (*none*) – Amplitude of current pulse

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters

return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Q10ConductanceScaling

`class` neuroml.nml.nml.`Q10ConductanceScaling`(*q10_factor: a Nml2Quantity_none (required) = None*, *experimental_temp: a Nml2Quantity_temperature (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Q10ConductanceScaling – A value for the conductance scaling which varies as a standard function of the difference between the current temperature, **temperature,** and the temperature at which the conductance was originally determined, **experimentalTemp**

Parameters

- **q10Factor** (*none*) –
- **experimentalTemp** (*temperature*) –

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to

- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Q10Settings

class neuroml.nml.nml.**Q10Settings**(*type: a NmlId (required) = None*, *fixed_q10: a Nml2Quantity_none (optional) = None*, *q10_factor: a Nml2Quantity_none (optional) = None*, *experimental_temp: a Nml2Quantity_temperature (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *GeneratedsSuper*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

RampGenerator

class neuroml.nml.nml.**RampGenerator**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *delay: a Nml2Quantity_time (required) = None*, *duration: a Nml2Quantity_time (required) = None*, *start_amplitude: a Nml2Quantity_current (required) = None*, *finish_amplitude: a Nml2Quantity_current (required) = None*, *baseline_amplitude: a Nml2Quantity_current (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

RampGenerator – Generates a ramping current after a time **delay,** for a fixed **duration.** During this time the current steadily changes from **startAmplitude** to **finishAmplitude.** Scaled by **weight,** if set

Parameters

- **delay** (*time*) – Delay before change in current. Current is baselineAmplitude prior to this.
- **duration** (*time*) – Duration for holding current at amplitude. Current is baselineAmplitude after delay + duration.
- **startAmplitude** (*current*) – Amplitude of linearly varying current at time delay
- **finishAmplitude** (*current*) – Amplitude of linearly varying current at time delay + duration
- **baselineAmplitude** (*current*) – Amplitude of current before time delay, and after time delay + duration

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

RampGeneratorDL

class neuroml.nml.nml.**RampGeneratorDL**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *delay: a Nml2Quantity_time (required) = None*, *duration: a Nml2Quantity_time (required) = None*, *start_amplitude: a Nml2Quantity_current (required) = None*, *finish_amplitude: a Nml2Quantity_current (required) = None*, *baseline_amplitude: a Nml2Quantity_current (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

RampGeneratorDL – Dimensionless equivalent of **rampGenerator** . Generates a ramping current after a time **delay,** for a fixed **duration.** During this time the dimensionless current steadily changes from **startAmplitude** to **finishAmplitude.** Scaled by **weight,** if set

Parameters

- **delay** (*time*) – Delay before change in current. Current is baselineAmplitude prior to this.
- **duration** (*time*) – Duration for holding current at amplitude. Current is baselineAmplitude after delay + duration.
- **startAmplitude** (*none*) – Amplitude of linearly varying current at time delay
- **finishAmplitude** (*none*) – Amplitude of linearly varying current at time delay + duration
- **baselineAmplitude** (*none*) – Amplitude of current before time delay, and after time delay + duration

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

RandomLayout

class neuroml.nml.nml.**RandomLayout**(*number: a nonNegativeInteger (optional) = None*, *regions: a NmlId (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

ReactionScheme

class neuroml.nml.nml.**ReactionScheme**(*id: a NmlId (required) = None*, *source: a string (required) = None*, *type: a string (required) = None*, *anytypeobjs_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Region

class neuroml.nml.nml.**Region**(*id: a NmlId (required) = None*, *spaces: a NmlId (optional) = None*, *anytypeobjs_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

Region – Initial attempt to specify 3D region for placing cells. Work in progress...

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child

elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Requirement

class neuroml.nml.nml.**Requirement**(*name: a string (required) = None*, *dimension: a string (required) = None*, *description: a string (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *NamedDimensionalType*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Resistivity

class neuroml.nml.nml.**Resistivity**(*value: a Nml2Quantity_resistivity (required) = None*, *segment_groups: a NmlId (optional) = 'all'*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

Resistivity – The resistivity, or specific axial resistance, of the cytoplasm

Parameters
value (*resistivity*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

validate_Nml2Quantity_resistivity(*value*)

validate_Nml2Quantity_resistivity_patterns_ =
[['^(-?([0-9]*(\\.[0-9]+)?)([eE]-?[0-9]+)?[\\s]*(ohm_cm|kohm_cm|ohm_m))$']]

ReverseTransition

class neuroml.nml.nml.**ReverseTransition**(*id: a NmlId (required) = None*, *from_: a NmlId (required) = None*, *to: a NmlId (required) = None*, *anytypeobjs_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

ReverseTransition – A reverse only **KSTransition** for a **gateKS** which specifies a **rate** (type **baseHHRate**) which follows one of the standard Hodgkin Huxley forms (e. g. **HHExpRate** , **HHSigmoidRate** , **HHExpLinearRate**

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Segment

class neuroml.nml.nml.**Segment**(*id: a NonNegativeInteger (required) = None*, *name: a string (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *parent: a SegmentParent (optional) = None*, *proximal: a Point3DWithDiam (optional) = None*, *distal: a Point3DWithDiam (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseNonNegativeIntegerId*

Segment – A segment defines the smallest unit within a possibly branching structure (**morphology**), such as a dendrite or axon. Its **id** should be a nonnegative integer (usually soma/root = 0). Its end points are given by

the **proximal** and **distal** points. The **proximal** point can be omitted, usually because it is the same as a point on the **parent** segment, see **proximal** for details. **parent** specifies the parent segment. The first segment of a **cell** (with no **parent**) usually represents the soma. The shape is normally a cylinder (radii of the **proximal** and **distal** equal, but positions different) or a conical frustum (radii and positions different). If the x, y, x positions of the **proximal** and **distal** are equal, the segment can be interpreted as a sphere, and in this case the radii of these points must be equal. NOTE: LEMS does not yet support multicompartmental modelling, so the Dynamics here is only appropriate for single compartment modelling.

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

`classmethod component_factory`(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

property length

Get the length of the segment.

Returns
length of the segment

Return type
float

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

property surface_area

Get the surface area of the segment.

Returns
surface area of segment

Return type
float

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

property volume

Get the volume of the segment.

Returns
volume of segment

Return type
float

SegmentEndPoint

class neuroml.nml.nml.**SegmentEndPoint**(*segments: a NonNegativeInteger (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SegmentGroup

class neuroml.nml.nml.**SegmentGroup**(*id: a NonNegativeInteger (required) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *members: list of Member(s) (optional) = None*, *includes: list of Include(s) (optional) = None*, *paths: list of Path(s) (optional) = None*, *sub_trees: list of SubTree(s) (optional) = None*, *inhomogeneous_parameters: list of InhomogeneousParameter(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

SegmentGroup – A method to describe a group of **segment** s in a **morphology** , e. g. soma_group, dendrite_group, axon_group. While a name is useful to describe the group, the **neuroLexId** attribute can be used to explicitly specify the meaning of the group, e. g. sao1044911821 for 'Neuronal Cell Body', sao1211023249 for 'Dendrite'. The **segment** s in this group can be specified as: a list of individual **member** segments; a **path** , all of the segments along which should be included; a **subTree** of the **cell** to include; other segmentGroups to **include** (so all segments from those get included here). An **inhomogeneousParameter** can be defined on the region of the cell specified by this group (see **variableParameter** for usage).

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)

- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SegmentParent

class neuroml.nml.nml.**SegmentParent**(*segments: a NonNegativeInteger (required) = None*, *fraction_along: a ZeroToOne (optional) = '1'*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SilentSynapse

class neuroml.nml.nml.**SilentSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseSynapse*

SilentSynapse – Dummy synapse which emits no current. Used as presynaptic endpoint for analog synaptic connection.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns

info string, or list of members or dict with members as keys and member values as values

Return type

str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters

return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SineGenerator

`class` neuroml.nml.nml.`SineGenerator`(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *delay: a Nml2Quantity_time (required) = None*, *phase: a Nml2Quantity_none (required) = None*, *duration: a Nml2Quantity_time (required) = None*, *amplitude: a Nml2Quantity_current (required) = None*, *period: a Nml2Quantity_time (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

SineGenerator – Generates a sinusoidally varying current after a time **delay,** for a fixed **duration.** The **period** and maximum **amplitude** of the current can be set as well as the **phase** at which to start. Scaled by **weight,** if set

Parameters

- **phase** (*none*) – Phase (between 0 and 2*pi) at which to start the varying current (i. e. at time given by delay)
- **delay** (*time*) – Delay before change in current. Current is zero prior to this.
- **duration** (*time*) – Duration for holding current at amplitude. Current is zero after delay + duration.
- **amplitude** (*current*) – Maximum amplitude of current
- **period** (*time*) – Time period of oscillation

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SineGeneratorDL

class neuroml.nml.nml.**SineGeneratorDL**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *delay: a Nml2Quantity_time (required) = None*, *phase: a Nml2Quantity_none (required) = None*, *duration: a Nml2Quantity_time (required) = None*, *amplitude: a Nml2Quantity_current (required) = None*, *period: a Nml2Quantity_time (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

SineGeneratorDL – Dimensionless equivalent of **sineGenerator** . Generates a sinusoidally varying current after a time **delay,** for a fixed **duration.** The **period** and maximum **amplitude** of the current can be set as well as the **phase** at which to start. Scaled by **weight,** if set

Parameters

- **phase** (*none*) – Phase (between 0 and 2*pi) at which to start the varying current (i. e. at time given by delay)
- **delay** (*time*) – Delay before change in current. Current is zero prior to this.
- **duration** (*time*) – Duration for holding current at amplitude. Current is zero after delay + duration.

- **amplitude** (*none*) – Maximum amplitude of current
- **period** (*time*) – Time period of oscillation

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids

can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Space

class neuroml.nml.nml.**Space**(*id: a NmlId (required) = None*, *based_on: a allowedSpaces (optional) = None*, *structure: a SpaceStructure (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to

- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SpaceStructure

class neuroml.nml.nml.**SpaceStructure**(*x_spacing: a float (optional) = None*, *y_spacing: a float (optional) = None*, *z_spacing: a float (optional) = None*, *x_start: a float (optional) = 0*, *y_start: a float (optional) = 0*, *z_start: a float (optional) = 0*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members

- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Species

class neuroml.nml.nml.**Species**(*id: a NmlId (required) = None*, *concentration_model: a NmlId (required) = None*, *ion: a NmlId (optional) = None*, *initial_concentration: a Nml2Quantity_concentration (required) = None*, *initial_ext_concentration: a Nml2Quantity_concentration (required) = None*, *segment_groups: a NmlId (optional) = 'all'*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

Species – Description of a chemical species identified by **ion,** which has internal, **concentration,** and external, **extConcentration** values for its concentration

Parameters

- **initialConcentration** (*concentration*) –
- **initialExtConcentration** (*concentration*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SpecificCapacitance

class neuroml.nml.nml.**SpecificCapacitance**(*value: a Nml2Quantity_specificCapacitance (required) = None*, *segment_groups: a NmlId (optional) = 'all'*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

SpecificCapacitance – Capacitance per unit area

Parameters
value (*specificCapacitance*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Spike

class neuroml.nml.nml.**Spike**(*id: a NonNegativeInteger (required) = None*, *time: a Nml2Quantity_time (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseNonNegativeIntegerId*

Spike – Emits a single spike at the specified **time**

Parameters
time (*time*) – Time at which to emit one spike event

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SpikeArray

class neuroml.nml.nml.**SpikeArray**(*id: a NonNegativeInteger (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *spikes: list of Spike(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

SpikeArray – Set of spike ComponentTypes, each emitting one spike at a certain time. Can be used to feed a predetermined spike train into a cell

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None

- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SpikeGenerator

class neuroml.nml.nml.**SpikeGenerator**(*id: a NonNegativeInteger (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *period: a Nml2Quantity_time (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

SpikeGenerator – Simple generator of spikes at a regular interval set by **period**

Parameters
period (*time*) – Time between spikes. The first spike will be emitted after this time.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SpikeGeneratorPoisson

class neuroml.nml.nml.**SpikeGeneratorPoisson**(*id: a NonNegativeInteger (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *average_rate: a Nml2Quantity_pertime (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

SpikeGeneratorPoisson – Generator of spikes whose ISI is distributed according to an exponential PDF with scale: 1 / **averageRate**

Parameters
averageRate (*per_time*) – The average rate at which spikes are emitted

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SpikeGeneratorRandom

class neuroml.nml.nml.**SpikeGeneratorRandom**(*id: a NonNegativeInteger (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *max_isi: a Nml2Quantity_time (required) = None*, *min_isi: a Nml2Quantity_time (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

SpikeGeneratorRandom – Generator of spikes with a random interspike interval of at least **minISI** and at most **maxISI**

Parameters

- **maxISI** (*time*) – Maximum interspike interval
- **minISI** (*time*) – Minimum interspike interval

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SpikeGeneratorRefPoisson

class neuroml.nml.nml.**SpikeGeneratorRefPoisson**(*id: a NonNegativeInteger (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *average_rate: a Nml2Quantity_pertime (required) = None*, *minimum_isi: a Nml2Quantity_time (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *SpikeGeneratorPoisson*

SpikeGeneratorRefPoisson – Generator of spikes whose ISI distribution is the maximum entropy distribution over [**minimumISI,** +infinity) with mean: 1 / **averageRate**

Parameters

- **minimumISI** (*time*) – The minimum interspike interval
- **averageRate** (*per_time*) – The average rate at which spikes are emitted

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SpikeSourcePoisson

class neuroml.nml.nml.**SpikeSourcePoisson**(*id: a NonNegativeInteger (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *start: a Nml2Quantity_time (required) = None*, *duration: a Nml2Quantity_time (required) = None*, *rate: a Nml2Quantity_pertime (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

SpikeSourcePoisson – Spike source, generating spikes according to a Poisson process.

Parameters

- **start** (*time*) –
- **duration** (*time*) –
- **rate** (*per_time*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)

- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SpikeThresh

class neuroml.nml.nml.**SpikeThresh**(*value: a Nml2Quantity_voltage (required) = None*, *segment_groups: a NmlId (optional) = 'all'*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

SpikeThresh – Membrane potential at which to emit a spiking event. Note, usually the spiking event will not be emitted again until the membrane potential has fallen below this value and rises again to cross it in a positive direction

Parameters
value (*voltage*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

Standalone

class neuroml.nml.nml.**Standalone**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

Standalone – Elements which can stand alone and be referenced by id, e.g. cell, morphology.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

StateVariable

class neuroml.nml.nml.**StateVariable**(*name: a string (required) = None*, *dimension: a string (required) = None*, *description: a string (optional) = None*, *exposure: a string (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *NamedDimensionalVariable*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously

- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *Member-Spec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SubTree

class neuroml.nml.nml.**SubTree**(*from_: a SegmentEndPoint (optional) = None*, *to: a SegmentEndPoint (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

SubTree – Include all the **segment** s distal to that specified by **from** in the **segmentGroup**

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

SynapticConnection

class neuroml.nml.nml.**SynapticConnection**(*neuro_lex_id: a NeuroLexId (optional) = None*, *from_: a string (required) = None*, *to: a string (required) = None*, *synapse: a string (required) = None*, *destination: a NmlId (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

SynapticConnection – Explicit event connection between named components, which gets processed via a new instance of a **synapse** component which is created on the target component

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

TauInfTransition

class neuroml.nml.nml.**TauInfTransition**(*id: a NmlId (required) = None*, *from_: a NmlId (required) = None*, *to: a NmlId (required) = None*, *steady_state: a HHVariable (required) = None*, *time_course: a HHTime (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Base*

TauInfTransition – KS Transition specified in terms of time constant **tau** and steady state **inf**

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

TimeDerivative

class neuroml.nml.nml.**TimeDerivative**(*variable: a string (required) = None*, *value: a string (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *GeneratedsSuper*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

TimedSynapticInput

class neuroml.nml.nml.**TimedSynapticInput**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *synapse: a NmlId (required) = None*, *spike_target: a string (required) = None*, *spikes: list of Spike(s) (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

TimedSynapticInput – Spike array connected to a single **synapse,** producing a current triggered by each **spike** in the array.

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided,

only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
`return_format` (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
`recursive` (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
`ValueError` – if component is invalid

TransientPoissonFiringSynapse

class neuroml.nml.nml.**TransientPoissonFiringSynapse**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *average_rate: a Nml2Quantity_pertime (required) = None*, *delay: a Nml2Quantity_time (required) = None*, *duration: a Nml2Quantity_time (required) = None*, *synapse: a string (required) = None*, *spike_target: a string (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

TransientPoissonFiringSynapse – Poisson spike generator firing at **averageRate** after a **delay** and for a **duration,** connected to single **synapse** that is triggered every time a spike is generated, providing an input current. Similar to ComponentType **poissonFiringSynapse** .

Parameters

- **averageRate** (*per_time*) –
- **delay** (*time*) –
- **duration** (*time*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

`parentinfo`(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

`validate`(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

UnstructuredLayout

class neuroml.nml.nml.**UnstructuredLayout**(*number: a nonNegativeInteger (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseWithoutId*

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

VariableParameter

class neuroml.nml.nml.**VariableParameter**(*parameter: a string (required) = None*, *segment_groups: a string (required) = None*, *inhomogeneous_value: a InhomogeneousValue (optional) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *GeneratedsSuper*

VariableParameter – Specifies a **parameter** (e. g. condDensity) which can vary its value across a **segmentGroup.** The value is calculated from **value** attribute of the **inhomogeneousValue** subelement. This element

is normally a child of **channelDensityNonUniform** , **channelDensityNonUniformNernst** or **channelDensityNonUniformGHK** and is used to calculate the value of the conductance, etc. which will vary on different parts of the cell. The **segmentGroup** specified here needs to define an **inhomogeneousParameter** (referenced from **inhomogeneousParameter** in the **inhomogeneousValue**), which calculates a **variable** (e. g. p) varying across the cell (e. g. based on the path length from soma), which is then used in the **value** attribute of the **inhomogeneousValue** (so for example condDensity = f(p))

`add`(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

`classmethod component_factory`(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

VoltageClamp

class neuroml.nml.nml.**VoltageClamp**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *delay: a Nml2Quantity_time (required) = None*, *duration: a Nml2Quantity_time (required) = None*, *target_voltage: a Nml2Quantity_voltage (required) = None*, *simple_series_resistance: a Nml2Quantity_resistance (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

VoltageClamp – Voltage clamp. Applies a variable current **i** to try to keep parent at **targetVoltage.** Not yet fully tested!!! Consider using voltageClampTriple!!

Parameters

- **delay** (*time*) – Delay before change in current. Current is zero prior to this.

- **duration** (*time*) – Duration for attempting to keep parent at targetVoltage. Current is zero after delay + duration.
- **targetVoltage** (*voltage*) – Current will be applied to try to get parent to this target voltage
- **simpleSeriesResistance** (*resistance*) – Current will be calculated by the difference in voltage between the target and parent, divided by this value

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.

- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

VoltageClampTriple

class neuroml.nml.nml.**VoltageClampTriple**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *active: a ZeroOrOne (required) = None*, *delay: a Nml2Quantity_time (required) = None*, *duration: a Nml2Quantity_time (required) = None*, *conditioning_voltage: a Nml2Quantity_voltage (required) = None*, *testing_voltage: a Nml2Quantity_voltage (required) = None*, *return_voltage: a Nml2Quantity_voltage (required) = None*, *simple_series_resistance: a Nml2Quantity_resistance (required) = None*, *gds_collector_=None*, ***kwargs_*)

Bases: *Standalone*

VoltageClampTriple – Voltage clamp with 3 clamp levels. Applies a variable current **i** (through **simpleSeriesResistance**) to try to keep parent cell at **conditioningVoltage** until time **delay, testingVoltage** until **delay** + **duration,** and **returnVoltage** afterwards. Only enabled if **active** = 1.

Parameters

- **active** (*none*) – Whether the voltage clamp is active (1) or inactive (0).
- **delay** (*time*) – Delay before switching from conditioningVoltage to testingVoltage.
- **duration** (*time*) – Duration to hold at testingVoltage.
- **conditioningVoltage** (*voltage*) – Target voltage before time delay
- **testingVoltage** (*voltage*) – Target voltage between times delay and delay + duration
- **returnVoltage** (*voltage*) – Target voltage after time duration
- **simpleSeriesResistance** (*resistance*) – Current will be calculated by the difference in voltage between the target and parent, divided by this value

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

basePyNNCell

class neuroml.nml.nml.**basePyNNCell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *cm: a float (required) = None*, *i_offset: a float (required) = None*, *tau_syn_E: a float (required) = None*, *tau_syn_I: a float (required) = None*, *v_init: a float (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *BaseCell*

basePyNNCell – Base type of any PyNN standard cell model. Note: membrane potential **v** has dimensions voltage, but all other parameters are dimensionless. This is to facilitate translation to and from PyNN scripts in Python, where these parameters have implicit units, see http://neuralensemble.org/trac/PyNN/wiki/StandardModels

Parameters

- **cm** (*none*) –
- **i_offset** (*none*) –
- **tau_syn_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_syn_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **v_init** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

basePyNNIaFCell

class neuroml.nml.nml.**basePyNNIaFCell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *cm: a float (required) = None*, *i_offset: a float (required) = None*, *tau_syn_E: a float (required) = None*, *tau_syn_I: a float (required) = None*, *v_init: a float (required) = None*, *tau_m: a float (required) = None*, *tau_refrac: a float (required) = None*, *v_reset: a float (required) = None*, *v_rest: a float (required) = None*, *v_thresh: a float (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *basePyNNCell*

basePyNNIaFCell – Base type of any PyNN standard integrate and fire model

Parameters

- **tau_refrac** (*none*) –
- **v_thresh** (*none*) –
- **tau_m** (*none*) –
- **v_rest** (*none*) –
- **v_reset** (*none*) –
- **cm** (*none*) –
- **i_offset** (*none*) –
- **tau_syn_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_syn_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **v_init** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj
the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns
new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

basePyNNIaFCondCell

class neuroml.nml.nml.**basePyNNIaFCondCell**(*id: a NmlId (required) = None*, *metaid: a MetaId (optional) = None*, *notes: a string (optional) = None*, *properties: list of Property(s) (optional) = None*, *annotation: a Annotation (optional) = None*, *neuro_lex_id: a NeuroLexId (optional) = None*, *cm: a float (required) = None*, *i_offset: a float (required) = None*, *tau_syn_E: a float (required) = None*, *tau_syn_I: a float (required) = None*, *v_init: a float (required) = None*, *tau_m: a float (required) = None*, *tau_refrac: a float (required) = None*, *v_reset: a float (required) = None*, *v_rest: a float (required) = None*, *v_thresh: a float (required) = None*, *e_rev_E: a float (required) = None*, *e_rev_I: a float (required) = None*, *extensiontype_=None*, *gds_collector_=None*, ***kwargs_*)

Bases: *basePyNNIaFCell*

basePyNNIaFCondCell – Base type of conductance based PyNN IaF cell models

Parameters

- **e_rev_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **e_rev_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_refrac** (*none*) –
- **v_thresh** (*none*) –
- **tau_m** (*none*) –
- **v_rest** (*none*) –
- **v_reset** (*none*) –
- **cm** (*none*) –
- **i_offset** (*none*) –
- **tau_syn_E** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **tau_syn_I** (*none*) – This parameter is never used in the NeuroML2 description of this cell! Any synapse producing a current can be placed on this cell
- **v_init** (*none*) –

add(*obj=None*, *hint=None*, *force=False*, *validate=True*, ***kwargs*)

Generic function to allow easy addition of a new member to a NeuroML object. Without arguments, when *obj=None*, it simply calls the *info()* method to provide the list of valid member types for the NeuroML class.

Please use the *info()* method directly for more information on the current contents of this component object.

When *obj* is given a string name of a NeuroML class ("NeuroMLDocument"), or the class itself (neuroml.NeuroMLDocument), a new object will be created of this type and added as a member to the calling (parent) component type object.

Parameters

- **obj** (*Object*) – member object or class type (neuroml.NeuroMLDocument) or name of class type ("NeuroMLDocument"), or None
- **hint** (*string*) – member name to add to when there are multiple members that *obj* can be added to
- **force** (*bool*) – boolean to force addition when an obj has already been added previously
- **validate** (*bool*) – validate component after adding (default: True)

Returns obj

the provided or created object

Raises

- **Exception** – if a member compatible to obj could not be found
- **Exception** – if multiple members can accept the object and no hint is provided.

classmethod component_factory(*component_type*, *validate=True*, ***kwargs*)

Factory function to create a NeuroML Component object.

Users can provide the name of the component as a string or the class variable, along with its named constructor arguments, and this function will create a new object of the Component and return it.

Users can use the *add()* helper function to further modify components

This factory runs two checks while creating the component object:

- that all arguments given do belong to the ComponentType (useful for caching typos)
- that the created component is valid NeuroML

It is therefore less error prone than creating Components directly using the ComponentType constructors.

It may be useful to disable validation when starting a model. The *validate* parameter can be set to False for this.

Parameters

- **component_type** (*str/type*) – component type to create component from: this can either be the name of the component as a string, e.g. "NeuroMLDocument", or it can be the class type itself: NeuroMLDocument. Note that when providing the class type, one will need to import it, e.g.: *import NeuroMLDocument*, to ensure that it is defined, whereas this will not be required when using the string.
- **validate** (*bool*) – toggle validation (default: True)
- **kwargs** (*named arguments*) – named arguments to be passed to ComponentType constructor

Returns

new Component (object) of provided ComponentType

Return type
object

Raises
ValueError – if validation/checks fail

info(*show_contents=False*, *return_format='string'*)

Provide information on NeuroML component.

This is useful to quickly check what members can go into a particular NeuroML class (which will match the Schema definitions). It lists these members and notes whether they are "single" type elements (Child elements) or "List" elements (Children elements). It will also note whether a member is optional or required.

To get a list of possible parents, use the *parentinfo()* method.

By default, this will only show the members, and not their contents. To see contents that have been set, use *show_contents=True*. This will not show empty/unset contents. To see all contents, set *show_contents=all*.

Note that not all members will have ids (since not all NeuroML2 ComponentTypes have ids). For members that do not have ids, the object reference is listed instead.

See http://www.davekuhlman.org/generateDS.html#user-methods for more information on the *MemberSpec_* class that generateDS uses.

Parameters

- **show_contents** (*bool or str*) – toggle to print out the contents of the members
- **return_format** (*str*) – select what format to return information in "string" (default), or "dict" or "list".

  If "dict" or "list" is provided, the information is returned as a dict/list instead of being printed. Note that if *show_contents* is *False*, only a list of members is available and will be returned even if "dict" is supplied. If *show_contents* is *True* or "all" but "list" is provided, only the list of members will be returned. If something other than "string", "list", or "dict" is provided, the string representation is returned and printed.

Returns
info string, or list of members or dict with members as keys and member values as values

Return type
str, list/dict

parentinfo(*return_format='string'*)

Show the list of possible parents.

This object can then be added to objects of the parents using the *add* method.

It is similar to the *info()* method. However, where in the *info()* method, it is possible to find the contents of members for a component (object) rather easily, it is not so easily possible to get all the objects that may refer to another object.

So, this will provide information on possible parents. It will not provide information on whether the components (objects) of the particular parent have already been instantiated and what their values are. The user should be able to gather this information easily by reading the sources.

Please also note that various component types in NeuroML take ids of components as parameters. For example, an *ExplicitInput* will take the id of a cell as its *target*, and the id of a *PulseGenerator* as *input*. However, these are string fields, and the cell/pulse generator classes do not currently know that their ids can be used in *ExplicitInput*. This information does not live in the XSD schema, and so cannot be obtained here either.

Parameters
return_format (*str*) – format in which to return information. If "string" (default), an information string is returned. If "list" or "dict", a list or dictionary is returned. The list will only contain the parent names, whereas the dict will also include the member of the parent that the component type matches to.

Returns
info string, or list of parents or dict with parents as keys and member information as values

Return type
str, list/dict

validate(*recursive=False*)

Validate the component.

Throws a Python *ValueError* if a the component is invalid. You can ignore this by using a *try .. except ValueError: pass* block.

Note: validating your NeuroML file against the schema, which pynml and jnml do, will also check this.

Note: that this is different from the *validate_* method, which does not validate inherited members.

Parameters
recursive (*bool*) – toggle recursive validation (default: False)

Returns
None

Return type
None

Raises
ValueError – if component is invalid

1.3.2 loaders Module

class neuroml.loaders.**ArrayMorphLoader**

Bases: object

classmethod load(*filepath*)

Right now this load method isn't done in a very nice way. TODO: Complete refactoring.

class neuroml.loaders.**NeuroMLHdf5Loader**

Bases: object

classmethod load(*src*, *optimized=False*)

class neuroml.loaders.**NeuroMLLoader**

Bases: object

classmethod load(*src*)

class neuroml.loaders.**SWCLoader**

Bases: object

WARNING: Class defunct

classmethod load_swc_single(*src*, *name=None*)

neuroml.loaders.**print_**(*text*, *verbose=True*)

neuroml.loaders.**read_neuroml2_file**(*nml2_file_name: str*, *include_includes: bool = False*, *verbose: bool = False*, *already_included: ~typing.Optional[list] = None*, *print_method: ~typing.Callable = <function print_>*, *optimized: bool = False*) → *NeuroMLDocument*

Read a NeuroML2 file into a NeuroMLDocument object

Parameters

- **nml2_file_name** (*str*) – name of NeuroML file to read
- **include_includes** (*bool*) – toggle whether Included files should also be loaded
- **verbose** (*bool*) – toggle verbose output
- **already_included** (*list*) – list of already included files
- **print_method** (*Callable*) – print function to use
- **optimised** (*bool*) – for optimised HDF5 NeuroML files

Returns

NeuroMLDoc object containing the read file

neuroml.loaders.**read_neuroml2_string**(*nml2_string: str*, *include_includes: bool = False*, *verbose: bool = False*, *already_included: list = []*, *print_method: ~typing.Callable = <function print_>*, *optimized: bool = False*, *base_path: ~typing.Optional[str] = None*) → *NeuroMLDocument*

Read a NeuroML2 string into a NeuroMLDocument object

Parameters

- **nml2_string** (*str*) – NeuroML string to load
- **include_includes** (*bool*) – toggle whether Included files should also be loaded
- **verbose** (*bool*) – toggle verbose output
- **already_included** (*list*) – list of already included files
- **print_method** (*Callable*) – print function to use
- **optimised** (*bool*) – for optimised HDF5 NeuroML files
- **base_path** (*str*) –

Returns

NeuroMLDoc object containing the model

1.3.3 writers Module

class neuroml.writers.**ArrayMorphWriter**

Bases: object

For now just testing a simple method which can write a morphology, not a NeuroMLDocument.

classmethod write(*data*, *filepath*)

class neuroml.writers.**NeuroMLHdf5Writer**

Bases: object

classmethod write(*nml_doc*, *h5_file_name*, *embed_xml=True*, *compress=True*)

class neuroml.writers.**NeuroMLWriter**

Bases: object

classmethod write(*nmldoc*, *file*, *close=True*)

Writes from NeuroMLDocument to nml file in future can implement from other types via chain of responsibility pattern.

1.3.4 utils Module

Utilities for checking generated code

neuroml.utils.**add_all_to_document**(*nml_doc_src: NeuroMLDocument*, *nml_doc_tgt: NeuroMLDocument*, *verbose: bool = False*) → None

Add all members of the source NeuroML document to the target NeuroML document.

Parameters

- **nml_doc_src** (NeuroMLDocument) – source NeuroML document to copy from
- **nml_doc_tgt** (NeuroMLDocument) – target NeuroML document to copy to
- **verbose** (*bool*) – control verbosity of working

Raises

Exception – if a member could not be copied.

neuroml.utils.**append_to_element**(*parent*, *child*)

Append a child element to a parent Component

Parameters

- **parent** (*Object*) – parent NeuroML component to add element to
- **child** (*Object*) – child NeuroML component to be added to parent

Raises

Exception – when the child could not be added to the parent

neuroml.utils.**component_factory**(*component_type: Union[str, type]*, *validate: bool = True*, ***kwargs: Any*) → Any

Factory function to create a NeuroML Component object.

Wrapper around the component_factory method that is present in each NeuroML component type class.

Please see *GeneratedsSuperSuper.component_factory* for more information.

neuroml.utils.**ctinfo**(*component_type*)

Provide information on any neuroml Component Type class.

This creates a new object (component) of the component type and call its info() method.

Parameters

component_type (*str or type*) – component type to print information for, either a string (the name) or the class itself

Returns

informatin string

Return type

str

neuroml.utils.**ctparentinfo**(*component_type*)

Provide information on the parentage of any NeuroML Component Type class.

This creates a new object (component) of the component type and call its parentinfo() method.

Parameters
component_type (*str or type*) – component type to print information for, either a string (the name) or the class itself

Returns
information string

Return type
str

neuroml.utils.**get_summary**(*nml_file_name: str*) → str

Get a summary of the given NeuroML file.

Parameters
nml_file_name (*str*) – name of NeuroML file to get summary of

Returns
summary of provided file

Return type
str

neuroml.utils.**has_segment_fraction_info**(*connections: list*) → bool

Check if connections include fraction information

Parameters
connections (*list*) – list of connection objects

Returns
True if connections include fragment information, otherwise False

Return type
Boolean

neuroml.utils.**is_valid_neuroml2**(*file_name: str*) → None

Check if a file is valid NeuroML2.

Parameters
file_name (*str*) – name of NeuroML file to check

Returns
True if file is valid, False if not.

Return type
Boolean

neuroml.utils.**main**()

neuroml.utils.**print_summary**(*nml_file_name: str*) → None

Print a summary of the NeuroML model in the given file.

Parameters
nml_file_name (*str*) – name of NeuroML file to print summary of

neuroml.utils.**validate_neuroml2**(*file_name: str*) → None

Validate a NeuroML document against the NeuroML schema specification.

Parameters
file_name (*str*) – name of NeuroML file to validate.

Raises
ValueError – if document is invalid

1.3.5 arraymorph Module

1.4 Examples

The examples in this section are intended to give in depth overviews of how to accomplish specific tasks with libNeuroML.

These examples are located in the neuroml/examples directory and can be tested to confirm they work by running the run_all.py script.

Examples

1.4.1 Creating a NeuroML morphology

```
"""
Example of connecting segments together to create a
multicompartmental model of a cell.
"""

import neuroml
import neuroml.writers as writers

p = neuroml.Point3DWithDiam(x=0, y=0, z=0, diameter=50)
d = neuroml.Point3DWithDiam(x=50, y=0, z=0, diameter=50)
soma = neuroml.Segment(proximal=p, distal=d)
soma.name = "Soma"
soma.id = 0
```

(continues on next page)

(continued from previous page)

```
# Make an axon with 100 compartments:

parent = neuroml.SegmentParent(segments=soma.id)
parent_segment = soma
axon_segments = []
seg_id = 1

for i in range(100):
    p = neuroml.Point3DWithDiam(
        x=parent_segment.distal.x,
        y=parent_segment.distal.y,
        z=parent_segment.distal.z,
        diameter=0.1,
    )

    d = neuroml.Point3DWithDiam(
        x=parent_segment.distal.x + 10,
        y=parent_segment.distal.y,
        z=parent_segment.distal.z,
        diameter=0.1,
    )

    axon_segment = neuroml.Segment(proximal=p, distal=d, parent=parent)

    axon_segment.id = seg_id

    axon_segment.name = "axon_segment_" + str(axon_segment.id)

    # now reset everything:
    parent = neuroml.SegmentParent(segments=axon_segment.id)
    parent_segment = axon_segment
    seg_id += 1

    axon_segments.append(axon_segment)

test_morphology = neuroml.Morphology()
test_morphology.segments.append(soma)
test_morphology.segments += axon_segments
test_morphology.id = "TestMorphology"

cell = neuroml.Cell()
cell.name = "TestCell"
cell.id = "TestCell"
cell.morphology = test_morphology


doc = neuroml.NeuroMLDocument(id="TestNeuroMLDocument")

doc.cells.append(cell)

nml_file = "tmp/testmorphwrite.nml"
```

(continues on next page)

(continued from previous page)

```
writers.NeuroMLWriter.write(doc, nml_file)

print("Written morphology file to: " + nml_file)

###### Validate the NeuroML ######

from neuroml.utils import validate_neuroml2

validate_neuroml2(nml_file)
```

1.4.2 Loading and modifying a file

```
"""
In this example an axon is built, a morphology is loaded, the axon is
then connected to the loadeed morphology.
"""

import neuroml
import neuroml.loaders as loaders
import neuroml.writers as writers

fn = "./test_files/Purk2M9s.nml"
doc = loaders.NeuroMLLoader.load(fn)
print("Loaded morphology file from: " + fn)

# get the parent segment:
parent_segment = doc.cells[0].morphology.segments[0]

parent = neuroml.SegmentParent(segments=parent_segment.id)

# make an axon:
seg_id = 5000  # need a way to get a unique id from a morphology
axon_segments = []
for i in range(10):
    p = neuroml.Point3DWithDiam(
        x=parent_segment.distal.x,
        y=parent_segment.distal.y,
        z=parent_segment.distal.z,
        diameter=0.1,
    )

    d = neuroml.Point3DWithDiam(
        x=parent_segment.distal.x + 10,
        y=parent_segment.distal.y,
        z=parent_segment.distal.z,
        diameter=0.1,
    )

    axon_segment = neuroml.Segment(proximal=p, distal=d, parent=parent)
```

(continues on next page)

(continued from previous page)

```
    axon_segment.id = seg_id

    axon_segment.name = "axon_segment_" + str(axon_segment.id)

    # now reset everything:
    parent = neuroml.SegmentParent(segments=axon_segment.id)
    parent_segment = axon_segment
    seg_id += 1

    axon_segments.append(axon_segment)

doc.cells[0].morphology.segments += axon_segments

nml_file = "./tmp/modified_morphology.nml"

writers.NeuroMLWriter.write(doc, nml_file)

print("Saved modified morphology file to: " + nml_file)


###### Validate the NeuroML ######

from neuroml.utils import validate_neuroml2

validate_neuroml2(nml_file)
```

1.4.3 Building a network

```
"""

Example to build a full spiking IaF network
through libNeuroML, save it as XML and validate it

"""

from neuroml import NeuroMLDocument
from neuroml import IafCell
from neuroml import Network
from neuroml import ExpOneSynapse
from neuroml import Population
from neuroml import PulseGenerator
from neuroml import ExplicitInput
from neuroml import SynapticConnection
import neuroml.writers as writers
from random import random


nml_doc = NeuroMLDocument(id="IafNet")

IafCell0 = IafCell(
```

(continues on next page)

(continued from previous page)

```
    id="iaf0",
    C="1.0 nF",
    thresh="-50mV",
    reset="-65mV",
    leak_conductance="10 nS",
    leak_reversal="-65mV",
)

nml_doc.iaf_cells.append(IafCell0)

IafCell1 = IafCell(
    id="iaf1",
    C="1.0 nF",
    thresh="-50mV",
    reset="-65mV",
    leak_conductance="20 nS",
    leak_reversal="-65mV",
)

nml_doc.iaf_cells.append(IafCell1)

syn0 = ExpOneSynapse(id="syn0", gbase="65nS", erev="0mV", tau_decay="3ms")

nml_doc.exp_one_synapses.append(syn0)

net = Network(id="IafNet")

nml_doc.networks.append(net)

size0 = 5
pop0 = Population(id="IafPop0", component=IafCell0.id, size=size0)

net.populations.append(pop0)

size1 = 5
pop1 = Population(id="IafPop1", component=IafCell0.id, size=size1)

net.populations.append(pop1)

prob_connection = 0.5

for pre in range(0, size0):

    pg = PulseGenerator(
        id="pulseGen_%i" % pre,
        delay="0ms",
        duration="100ms",
        amplitude="%f nA" % (0.1 * random()),
    )

    nml_doc.pulse_generators.append(pg)
```

(continues on next page)

(continued from previous page)

```
    exp_input = ExplicitInput(target="%s[%i]" % (pop0.id, pre), input=pg.id)

    net.explicit_inputs.append(exp_input)

    for post in range(0, size1):
        # fromxx is used since from is Python keyword
        if random() <= prob_connection:
            syn = SynapticConnection(
                from_="%s[%i]" % (pop0.id, pre),
                synapse=syn0.id,
                to="%s[%i]" % (pop1.id, post),
            )
            net.synaptic_connections.append(syn)

nml_file = "tmp/testnet.nml"
writers.NeuroMLWriter.write(nml_doc, nml_file)


print("Written network file to: " + nml_file)


###### Validate the NeuroML ######

from neuroml.utils import validate_neuroml2

validate_neuroml2(nml_file)
```

1.4.4 Building a 3D network

```
"""

Example to build a full spiking IaF network throught libNeuroML & save it as XML &␣
↪validate it

"""

from neuroml import NeuroMLDocument
from neuroml import Network
from neuroml import ExpOneSynapse
from neuroml import Population
from neuroml import Property
from neuroml import Cell
from neuroml import Location
from neuroml import Instance
from neuroml import Morphology
from neuroml import Point3DWithDiam
from neuroml import Segment
from neuroml import SegmentParent
from neuroml import Projection
from neuroml import Connection
```

(continues on next page)

(continued from previous page)

```
import neuroml.writers as writers
from random import random

soma_diam = 10
soma_len = 10
dend_diam = 2
dend_len = 10
dend_num = 10


def generateRandomMorphology():

    morphology = Morphology()

    p = Point3DWithDiam(x=0, y=0, z=0, diameter=soma_diam)
    d = Point3DWithDiam(x=soma_len, y=0, z=0, diameter=soma_diam)
    soma = Segment(proximal=p, distal=d, name="Soma", id=0)

    morphology.segments.append(soma)
    parent_seg = soma

    for dend_id in range(0, dend_num):

        p = Point3DWithDiam(x=d.x, y=d.y, z=d.z, diameter=dend_diam)
        d = Point3DWithDiam(x=p.x, y=p.y + dend_len, z=p.z, diameter=dend_diam)
        dend = Segment(proximal=p, distal=d, name="Dend_%i" % dend_id, id=1 + dend_id)
        dend.parent = SegmentParent(segments=parent_seg.id)
        parent_seg = dend

        morphology.segments.append(dend)

    morphology.id = "TestMorphology"

    return morphology


def run():

    cell_num = 10
    x_size = 500
    y_size = 500
    z_size = 500

    nml_doc = NeuroMLDocument(id="Net3DExample")

    syn0 = ExpOneSynapse(id="syn0", gbase="65nS", erev="0mV", tau_decay="3ms")
    nml_doc.exp_one_synapses.append(syn0)

    net = Network(id="Net3D")
    nml_doc.networks.append(net)
```

(continues on next page)

(continued from previous page)

```
    proj_count = 0
    # conn_count = 0

    for cell_id in range(0, cell_num):

        cell = Cell(id="Cell_%i" % cell_id)

        cell.morphology = generateRandomMorphology()

        nml_doc.cells.append(cell)

        pop = Population(
            id="Pop_%i" % cell_id, component=cell.id, type="populationList"
        )
        net.populations.append(pop)
        pop.properties.append(Property(tag="color", value="1 0 0"))

        inst = Instance(id="0")
        pop.instances.append(inst)

        inst.location = Location(
            x=str(x_size * random()), y=str(y_size * random()), z=str(z_size * random())
        )

        prob_connection = 0.5
        for post in range(0, cell_num):
            if post is not cell_id and random() <= prob_connection:

                from_pop = "Pop_%i" % cell_id
                to_pop = "Pop_%i" % post

                pre_seg_id = 0
                post_seg_id = 1

                projection = Projection(
                    id="Proj_%i" % proj_count,
                    presynaptic_population=from_pop,
                    postsynaptic_population=to_pop,
                    synapse=syn0.id,
                )
                net.projections.append(projection)
                connection = Connection(
                    id=proj_count,
                    pre_cell_id="%s[%i]" % (from_pop, 0),
                    pre_segment_id=pre_seg_id,
                    pre_fraction_along=random(),
                    post_cell_id="%s[%i]" % (to_pop, 0),
                    post_segment_id=post_seg_id,
                    post_fraction_along=random(),
                )

                projection.connections.append(connection)
```

(continues on next page)

(continued from previous page)

```
                proj_count += 1
                # net.synaptic_connections.append(SynapticConnection(from_="%s[%i]"
↪%(from_pop,0),  to="%s[%i]"%(to_pop,0)))

    #######   Write to file  ######

    nml_file = "tmp/net3d.nml"
    writers.NeuroMLWriter.write(nml_doc, nml_file)

    print("Written network file to: " + nml_file)

    ###### Validate the NeuroML ######

    from neuroml.utils import validate_neuroml2

    validate_neuroml2(nml_file)


run()
```

1.4.5 Ion channels

```
"""
Generating a Hodgkin-Huxley Ion Channel and writing it to NeuroML
"""

import neuroml
import neuroml.writers as writers

chan = neuroml.IonChannelHH(
    id="na",
    conductance="10pS",
    species="na",
    notes="This is an example voltage-gated Na channel",
)

m_gate = neuroml.GateHHRates(id="m", instances="3")
h_gate = neuroml.GateHHRates(id="h", instances="1")

m_gate.forward_rate = neuroml.HHRate(
    type="HHExpRate", rate="0.07per_ms", midpoint="-65mV", scale="-20mV"
)

m_gate.reverse_rate = neuroml.HHRate(
    type="HHSigmoidRate", rate="1per_ms", midpoint="-35mV", scale="10mV"
)

h_gate.forward_rate = neuroml.HHRate(
    type="HHExpLinearRate", rate="0.1per_ms", midpoint="-55mV", scale="10mV"
)
```

(continues on next page)

(continued from previous page)

```
h_gate.reverse_rate = neuroml.HHRate(
    type="HHExpRate", rate="0.125per_ms", midpoint="-65mV", scale="-80mV"
)

chan.gate_hh_rates.append(m_gate)
chan.gate_hh_rates.append(h_gate)

doc = neuroml.NeuroMLDocument()
doc.ion_channel_hhs.append(chan)

doc.id = "ChannelMLDemo"

nml_file = "./tmp/ionChannelTest.xml"
writers.NeuroMLWriter.write(doc, nml_file)

print("Written channel file to: " + nml_file)


###### Validate the NeuroML ######

from neuroml.utils import validate_neuroml2

validate_neuroml2(nml_file)
```

1.4.6 PyNN models

```
"""

Example to build a PyNN based network

"""

from neuroml import NeuroMLDocument
from neuroml import *
import neuroml.writers as writers
from random import random


########################   Build the network   ##################################

nml_doc = NeuroMLDocument(id="IafNet")


pynn0 = IF_curr_alpha(
    id="IF_curr_alpha_pop_IF_curr_alpha",
    cm="1.0",
    i_offset="0.9",
    tau_m="20.0",
    tau_refrac="10.0",
```

(continues on next page)

(continued from previous page)

```
    tau_syn_E="0.5",
    tau_syn_I="0.5",
    v_init="-65",
    v_reset="-62.0",
    v_rest="-65.0",
    v_thresh="-52.0",
)
nml_doc.IF_curr_alpha.append(pynn0)

pynn1 = HH_cond_exp(
    id="HH_cond_exp_pop_HH_cond_exp",
    cm="0.2",
    e_rev_E="0.0",
    e_rev_I="-80.0",
    e_rev_K="-90.0",
    e_rev_Na="50.0",
    e_rev_leak="-65.0",
    g_leak="0.01",
    gbar_K="6.0",
    gbar_Na="20.0",
    i_offset="0.2",
    tau_syn_E="0.2",
    tau_syn_I="2.0",
    v_init="-65",
    v_offset="-63.0",
)
nml_doc.HH_cond_exp.append(pynn1)

pynnSynn0 = ExpCondSynapse(id="ps1", tau_syn="5", e_rev="0")
nml_doc.exp_cond_synapses.append(pynnSynn0)

nml_file = "tmp/pynn_network.xml"
writers.NeuroMLWriter.write(nml_doc, nml_file)
print("Saved to: " + nml_file)


###### Validate the NeuroML ######

from neuroml.utils import validate_neuroml2

validate_neuroml2(nml_file)
```

1.4.7 Synapses

```
"""

Example to create a file with multiple synapse types

"""

from neuroml import NeuroMLDocument
from neuroml import *
import neuroml.writers as writers
from random import random


nml_doc = NeuroMLDocument(id="SomeSynapses")

expOneSyn0 = ExpOneSynapse(id="ampa", tau_decay="5ms", gbase="1nS", erev="0mV")
nml_doc.exp_one_synapses.append(expOneSyn0)

expTwoSyn0 = ExpTwoSynapse(
    id="gaba", tau_decay="12ms", tau_rise="3ms", gbase="1nS", erev="-70mV"
)
nml_doc.exp_two_synapses.append(expTwoSyn0)

bpSyn = BlockingPlasticSynapse(
    id="blockStpSynDep", gbase="1nS", erev="0mV", tau_rise="0.1ms", tau_decay="2ms"
)
bpSyn.notes = "This is a note"
bpSyn.plasticity_mechanism = PlasticityMechanism(
    type="tsodyksMarkramDepMechanism", init_release_prob="0.5", tau_rec="120 ms"
)
bpSyn.block_mechanism = BlockMechanism(
    type="voltageConcDepBlockMechanism",
    species="mg",
    block_concentration="1.2 mM",
    scaling_conc="1.920544 mM",
    scaling_volt="16.129 mV",
)

nml_doc.blocking_plastic_synapses.append(bpSyn)


nml_file = "tmp/synapses.xml"
writers.NeuroMLWriter.write(nml_doc, nml_file)
print("Saved to: " + nml_file)


###### Validate the NeuroML ######

from neuroml.utils import validate_neuroml2

validate_neuroml2(nml_file)
```

1.4.8 Working with arraymorphs

```
"""
Example of connecting segments together to create a
multicompartmental model of a cell.

In this case ArrayMorphology will be used rather than
Morphology - demonstrating its similarity and
ability to save in HDF5 format
"""

import neuroml
import neuroml.writers as writers
import neuroml.arraymorph as am

p = neuroml.Point3DWithDiam(x=0, y=0, z=0, diameter=50)
d = neuroml.Point3DWithDiam(x=50, y=0, z=0, diameter=50)
soma = neuroml.Segment(proximal=p, distal=d)
soma.name = "Soma"
soma.id = 0

# now make an axon with 100 compartments:

parent = neuroml.SegmentParent(segments=soma.id)
parent_segment = soma
axon_segments = []
seg_id = 1
for i in range(100):
    p = neuroml.Point3DWithDiam(
        x=parent_segment.distal.x,
        y=parent_segment.distal.y,
        z=parent_segment.distal.z,
        diameter=0.1,
    )

    d = neuroml.Point3DWithDiam(
        x=parent_segment.distal.x + 10,
        y=parent_segment.distal.y,
        z=parent_segment.distal.z,
        diameter=0.1,
    )

    axon_segment = neuroml.Segment(proximal=p, distal=d, parent=parent)

    axon_segment.id = seg_id

    axon_segment.name = "axon_segment_" + str(axon_segment.id)

    # now reset everything:
    parent = neuroml.SegmentParent(segments=axon_segment.id)
    parent_segment = axon_segment
    seg_id += 1
```

(continues on next page)

(continued from previous page)

```
    axon_segments.append(axon_segment)

test_morphology = am.ArrayMorphology()
test_morphology.segments.append(soma)
test_morphology.segments += axon_segments
test_morphology.id = "TestMorphology"

cell = neuroml.Cell()
cell.name = "TestCell"
cell.id = "TestCell"
cell.morphology = test_morphology


doc = neuroml.NeuroMLDocument()
# doc.name = "Test neuroML document"

doc.cells.append(cell)
doc.id = "TestNeuroMLDocument"

nml_file = "tmp/arraymorph.nml"

writers.NeuroMLWriter.write(doc, nml_file)

print("Written morphology file to: " + nml_file)

###### Validate the NeuroML ######

from neuroml.utils import validate_neuroml2

validate_neuroml2(nml_file)
```

1.4.9 Working with Izhikevich Cells

These examples were kindly contributed by Steve Marsh

```
# from neuroml import NeuroMLDocument
from neuroml import IzhikevichCell
from neuroml.loaders import NeuroMLLoader
from neuroml.utils import validate_neuroml2


def load_izhikevich(filename="./test_files/SingleIzhikevich.nml"):
    nml_filename = filename
    validate_neuroml2(nml_filename)
    nml_doc = NeuroMLLoader.load(nml_filename)

    iz_cells = nml_doc.izhikevich_cells
    for i, iz in enumerate(iz_cells):
        if isinstance(iz, IzhikevichCell):
            neuron_string = "%d %s %s %s %s %s (%s)" % (
                i,
```

(continues on next page)

(continued from previous page)

```
                iz.v0,
                iz.a,
                iz.b,
                iz.c,
                iz.d,
                iz.id,
            )
            print(neuron_string)
        else:
            print("Error: Cell %d is not an IzhikevichCell" % i)


load_izhikevich()
```

```
from neuroml import NeuroMLDocument
from neuroml import IzhikevichCell
from neuroml.writers import NeuroMLWriter
from neuroml.utils import validate_neuroml2


def write_izhikevich(filename="./tmp/SingleIzhikevich_test.nml"):
    nml_doc = NeuroMLDocument(id="SingleIzhikevich")
    nml_filename = filename

    iz0 = IzhikevichCell(
        id="iz0", v0="-70mV", thresh="30mV", a="0.02", b="0.2", c="-65.0", d="6"
    )

    nml_doc.izhikevich_cells.append(iz0)

    NeuroMLWriter.write(nml_doc, nml_filename)
    validate_neuroml2(nml_filename)


write_izhikevich()
```

1.5 References

CHAPTER

TWO

CONTRIBUTING

2.1 How to contribute

libNeuroML development happens on GitHub, so you will need a GitHub account to contribute to the repository. Contributions are made using the standard Pull Request workflow.

2.1.1 Setting up

Please take a look at the GitHub documentation here: http://help.github.com/fork-a-repo/

To begin, please fork the repo on the GitHub website. You should now have a libNeuroML under you username. Next, we clone our fork to get a local copy on our computer:

```
git clone git@github.com:_username_/libNeuroML.git
```

While not necessary, it is good practice to add the upstream repository as a remote that you will follow:

```
cd libNeuroML
git remote add upstream https://github.com/NeuralEnsemble/libNeuroML.git
git fetch upstream
```

You can check which branch are you following doing:

```
git branch -a
```

You should have something like:

```
git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/upstream/master
```

2.1.2 Sync with upstream

Before starting to do some work, please check to see that you have the latest copy of the sources in your local repository:

```
git fetch upstream
git checkout development
git merge upstream/development
```

2.1.3 Working locally on a dedicated branch

Now that we have a fork, we can start making our changes to the source code. The best way to do it is to create a branch with a descriptive name to indicate what are you working on. Generally, your will branch off from the upstream *development* branch, which will contain the latest code.

For example, just for the sake of this guide, I'm going to work on issue #2.

```
git checkout development
git checkout -b fix-2
```

We can work in this branch, and make as many commits as we need to:

```
# hack hack hack
git commit -am "some decent commit message here"
```

Once we have finished working, we can push the branch online to our fork:

```
git push origin fix-2
```

We can then open a pull-request to merge our `fix-2` branch into `upstream/development`. If your code is not ready to be included, you can update the code on your branch and any more commits you add there will be added to the Pull Request. Members of the libNeuroML development team will then discuss your changes with you, perhaps suggest tweaks, and then merge it when ready.

2.1.4 Continuous integration

libNeuroML uses continuous integration (Wikipedia). Each commit to the master or development branches is tested, along with all commits to pull requests. The latest status of the continuous integration tests can be seen here on GitHub Actions.

2.1.5 Release process

libNeuroML is part of the official NeuroML release cycle. When a new libNeuroML release is ready the following needs to happen:

- Update version number in setup.py
- update version number in doc/conf.py
- update release number in doc/conf.py (same as version number)
- update changelog in README.md
- merge development branch with master (This should happen via pull request - do not do the merge yourself even if you are an owner of the repository.

- push latest release to PyPi

More information on the NeuroML release process can be found on the NeuroML documentation page.

2.2 Regenerating documentation

Please create a virtual environment and use the *requirements.txt* file to install the necessary bits.

In most cases, running *make html* should be sufficient to regenerate the documentation. However, if any changes to *nml.py* have been made, the *nml-core-docs.py* file in the *helpers* directory will also need to be run. This script manually adds each class from *nml.py* to the documentation as a sub-section using the *autoclass* sphinx directive instead of the *automodule* directive which does not allow us to do this.

2.3 Implementation of XML bindings for libNeuroML

The GenerateDS Python package is used to automatically generate the NeuroML XML-bindings in libNeuroML from the NeuroML Schema. This technique can be utilized for any XML Schema and is outlined in this section. The addition of helper methods and enforcement of correct naming conventions is also described. For more detail on how Python bindings for XML are generated, the reader is directed to the GenerateDS and libNeuroML documentation. In the following subsections it is assumed that all commands are executed in a top level directory nml and that GenerateDS is installed. It should be noted that enforcement of naming conventions and addition of helper methods are not required by GenerateDS and default values may be used.

2.3.1 Correct naming conventions

A module named generateds_config.py is placed in the nml directory. This module contains a Python dictionary called NameTable which maps the original names specified in the XML Schema to user-specified ones. The NameTable dictionary can be defined explicitly or generated programmatically, for example using regular expressions.

2.3.2 Addition of helper methods

Helper methods associated with a class can be added to a Python module as string objects. In the case of libNeuroML the module is called helper_methods.py. The precise implementation details are esoteric and the user is referred to the GenerateDS documentation for details of how this functionality is implemented.

2.3.3 Generation of bindings

Once generateds_config.py and a helper methods module are present in the nml directory a valid XML Schema is required by GenerateDS. The following command generates the nml.py module which contains the XML-bindings:

```
$ generateDS.py -o nml.py --use-getter-setter=none --user-methods=helper_methods NeuroML_
↪v2beta1.xsd
```

The -o flag sets the file which the module containing the bindings is to be written to. The –use-getter-setter=none option disables getters and setters for class attributes. The –user-methods flag indicates the name of the helper methods module (See section “Addition of helper methods”). The final parameter (NeuroML_v2beta1.xsd) is the name of the XML Schema used for generating the bindings.

2.4 Multicompartmental Python API Meeting

2.4.1 Organisation

Dates: 25 & 26 June 2012

Location: Room 336, Rockefeller building, UCL, London

Attendees: Sandra Berger, Andrew Davison, Padraig Gleeson, Mike Hull, Steve Marsh, Michele Mattioni, Eugenio Piasini, Mike Vella

Sponsors: This meeting was generously supported by the INCF Multi Scale Modelling Program.

2.4.2 Minutes

Agreeing on terminology (segments, etc.) & scope

A discussion on the definitions of the key terms Node, Segment and Section is here, and was the basis for discussions on these definitions at the meeting:

Nodes, Segments and Sections

Agreements

The Python libNeuroML API will use Node as a key building block for morphologies.

Segment is agreed on as the basis for defining morphologies in NeuroML and will be a top level object in libNeuroML, where it will be the part of a neurite between two Nodes (proximal & distal).

Segment Group will be the basis for the grouping of these, and will be used to define dendrites, axons, etc.

Section is a term for the cable-like building block in NEURON, and will not be formally used in NeuroML or libNeuroML.

There was a discussion on whether it would be useful to be able to include this concept "by the back door" to enable lossless import & export of morphologies from NEURON. Padraig's proposal was to add an attribute (e.g. primary) to the segmentGroup element to flag a core set of non overlapping segmentGroups, which are continuous (all children are connected to distal point of parent) which would correspond to the old "cable" concept in NeuroML v1.x.

There was much discussion on the usefulness of this concept and whether it should be a different element/object in the API from segmentGroup. The outcome was not fully resolved, but as a first test of this concept, Padraig will add the new attribute to NeuroML, Mike V will add a flag (boolean?) to the API, and at a later point, when the API begins to interact with native simulators, we can reevaluate the usefulness of the term.

Mike Vella's current implementation

This is under development at: https://github.com/NeuralEnsemble/libNeuroML/tree/master/neuroml

Mike will continue on this (almost) full time for the next 2 months.

Following the meeting, he will perform a refactoring operation on the code base to better reflect the names used in NeuroML, e.g.

```
neuroml_doc

    cells

        morphology # not entirely sure how this works- contains segment groups and is itself
        a segment group?
```

segments

segment_groups

segment_groups

biophysical_properties

notes

morphologies

networks

point currents

ion channels

synapses

extracellular properties

It was also decided that certain SegmentGroup names should have reserved names in libNeuroML, the exact implementation of this is undecided:

Segment groups with reserved names:

soma_group
axon_group
apical_dendrite_group
basal_dendrite_group

It was also decided that a segment should only be able to connect to the root of a morphology, the syntax should be something along the lines of:

segment can only connect to root of a morphology

connect syntax examples:

morph2.attach(2,cell2,0.5) (default frac along = None)

and:

morph[2].attach(cell2,0.5)

Mike V was asked to add a clone method to a morphology.

It was decided that fraction_along should be a property of segment.

The syntax for segment groups should be as follows: group=morph.segment_groups['axon_group'] (in connect merge groups should be false by default - throw an exception, tell the user setting merge_groups = True or rename group will fix this)

This was a subject of great debate and has not been completely settled.

Morphforge latest developments

Mike Hull gave a brief overview of the latest developments with Morphforge:

https://github.com/mikehulluk/morphforge

He pointed out that it's still undergoing refactoring, but it can be used by other interested parties, and there is detailed documentation online regarding installation, examples, etc.

Neuronvisio latest developments

Michele Mattioni gave a status update on Neuronvisio:

http://neuronvisio.org

The application has been closely linked to the NEURON simulator but hopefully use of libNeuroML will allow it to be used independently of NEURON.

Michele showed Neuronvisio's native HDF5 format as just one possible way to encode model structure + simulation results: https://github.com/NeuralEnsemble/libNeuroML/blob/master/hdf5Examples/Neuronvisio_medium_cell_example_10ms.h5

Current Python & NeuroML support in MOOSE

A Skype call/Google Hangout was held on Tues at 9:30 to get an update from Bangalore.

The slides from this discussion are here:

https://github.com/NeuralEnsemble/libNeuroML/blob/master/doc/2012_06_26_neuroml_with_pymoose.pdf

As outlined there there are a number of areas in which MOOSE and Moogli import/export NeuroML version 1.x. A number of issues and desired features missing in v1.x were highlighted, most of which are implemented or planned for NeuroML v2.0.

There was general enthusiasm about the libNeuroML project, and it was felt that MOOSE should eventually transition to using libNeuroML to import NeuroML models. This will happen in parallel with updating of the MOOSE PyNN implementation.

The MOOSE developers were also keen to see how the new ComponentTypes in NeuroML 2 will map to inbuilt objects in MOOSE (e.g. Integrate-and-Fire neurons, Markov channel, Izhikevich). They will add simple examples to the latest MOOSE code to demonstrate their current implementation and discussion can continue on the mailing lists.

Saving to & loading from XML

There was not any detailed discussion on the various strategies for reading/saving XML in Python.

Padraig's suggestion based on generateDS.py: https://github.com/NeuralEnsemble/libNeuroML/tree/master/ideas/padraig/generatedFromV2Schema produces a very big file, which while usable as an API, e.g. see:

https://github.com/NeuralEnsemble/libNeuroML/blob/master/hhExample/hh_NEUROML2.py

could do with a lot of refactoring. It was felt that a version of this with a very efficient description of morphologies (and network instances) based on the current work of Mike V is the way forward.

Storing simulation data as HDF5

The examples at: https://github.com/NeuralEnsemble/libNeuroML/tree/master/hdf5Examples have been updated.

The long term aim would be to arrive at a common format here that can be saved by simulators and that visualisation packages like Moogli and Neuronvisio can read and display. This may be based on Neo: http://packages.python.org/neo/, but that package's current lack of ability to deal with data with nonuniform time points (e.g. produced by variable time step simulations) may be a limiting factor.

General PyNN & NeuroML v2.0 interoperability

There was agreement that libNeuroML will form the basis of the multicompartmental neuron support in PyNN. The extra functionality needed to interact with simulators is currently termed "Pyramidal", but this will eventually be fully merged into PyNN.

http://neuralensemble.org/trac/PyNN http://www.neuroml.org/NeuroML2CoreTypes/PyNN.html http://www.neuroml.org/pynn.php

2.5 Nodes, Segments and Sections

An attempt to clarify these interrelated terms used in describing morphologies. Names in **bold type** are used for elements of the NeuroML object model.

2.5.1 Nodes

A node is a 3D point with diameter information which forms the basis for 3D morphological reconstructions.

These nodes (or points) are the fundamental building blocks in the SWC and Neurolucida formats. This method of description is based on the assumption that each node is physically connected to another node.

2.5.2 Segments

A **segment** (according to NeuroML v1&2) is a part of a neuronal tree between two 3D points with diameters (**proximal** & **distal**). The term node isn't used in NeuroML but the above description describes perfectly well the **proximal** & **distal** points. Cell **morphology** elements consist of lists of **segments** (each with unique integer id, and optional name).

All segments, apart from the root segment, have a **parent** segment. If the **proximal** point of the segment is not specified, the **distal** point of the parent segment is used for the **proximal** point of the child.

A special case is defined where **proximal** == **distal**, and the **segment** is assumed to be a sphere at that location with the specified diameter.

Segments can be grouped into **segmentGroups** in NeuroML v2.0. These can be used to specify "apical_dendrites", "axon_group", etc., which in turn can be used for placing channels on the cell.

An example of a NeuroML v2.0 cell is here.

libNeuroML will allow low level access to create and modify morphologies by handling nodes. Segments will also be top level objects in the API. The XML serialisation will only specify **segments** with **proximal** & **distal** points, but the HDF5 version may have an efficient serialisation of nodes & segments.

2.5.3 Sections

The concept of section is fundamentally important in NEURON. A section in this simulator is an unbranched cable which can have multiple 3D points outlining the structure of a neurite in 3D. These points are used to determine the surface area along the section. NEURON can vary the spatial discretisation of the neurite by varying the "nseg" value of the section, e.g. a section with 20 3D points and nseg =4 will be split into 4 parts of equal length for simulating (as isopotential compartments), with the surface area (and so total channel conductance) of each determined by the set of 3D points in that part.

There was a similar concept to this in NeuroML v1.x, the **cable**. Each **segment** had an attribute for the cable id, and these were used for mapping to and from NEURON. Cables were unbranched, and so all segments after the first in the cable only had distal points, see this example.

The cable concept was removed in NeuroML v2.0, as this is was seen as imposing concepts from compartmental modelling on the basic morphological descriptions of cells. There is only a **segmentGroup** element for grouping segments, though a **segment** can belong to multiple **segmentGroups**, which don't need to be unbranched (unlike **cables**). There may need to be a new attribute in **segmentGroup** (e.g. primary or unbranched or cable="true") which defines a nonoverlapping set of unbranched segmentGroups, which can be used as the basis for sections in any parsing application which is interested in them, or be ignored by any other application.

In libNeuroML, a section-like concept can be added at API level, to facilitate building cells, to facilitate import/export to/from simulators supporting this concept, and to serve as a basis for recompartmentalisation of cells.

2.5.4 Issues

Dendrites in space

One major issue to address is that in many neuronal reconstructions, the soma is not included (or perhaps just an outline of the soma is given), only the dendrites are. These dendrites' 3D start points are on the edge of the soma membrane "floating in space". Normal procedure for a modeller in this case is to create a spherical soma at this central point and electrically attach the dendrites to the centre of this.

In this case (and many others) the physical location of the start of the child segments do not correspond to the electrical (or logical) connection point on the parent. This has advantages and disadvantages:

(+) It allows the real 3D points of the neuronal reconstruction to be retained (useful for visualisation)

(-) This is not unambiguously captured in the simplest morphological formats like SWC, which assume physical connectivity between nodes/points

This scenario is supported in NeuroML v1&2, where a child **segment** has the option to redefine its start point (by adding a **proximal**) with the child <-> parent relationship defining the electrical connection. This allows lossless import & export from NEURON and removes the ambiguity of more compact formats like SWC and Neurolucida.

Connections mid segment

Another option for electrical connections (also influences by NEURON sections) is the ability for **segments** to (electrically/logically) connect to a point inside a **segment**. This is specified by adding a fractionAlong attribute to the **parent** element, i.e.

```
<parent segment="2" fractionAlong="0.5"/>
```

This is not possible in a node based format, but represents a logically consistent description of what the modeller wants.

What to do?

Two options are available then for a serialisation format or API: should it try to support all of these scenarios, or try to enforce "best practice"?

PG: I'd argue for the first approach, as it retains as much as possible of what the original reconstructor/simulator specified. An API which enforces a policy when it encounters a non optimal morphology (e.g. moving all dendrites to connection points, inserting new nodes) will alter the original data in perhaps unintended ways, and that information will be lost by subsequent readers. It should be up to each parsing application to decide what to do with the extra information when it reads in a file.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[VCC+14] Michael Vella, Robert C. Cannon, Sharon Crook, Andrew P. Davison, Gautham Ganapathy, Hugh P. C. Robinson, R. Angus Silver, and Padraig Gleeson. Libneuroml and pylems: using python to combine procedural and declarative modeling approaches in computational neuroscience. *Frontiers in neuroinformatics*, 8:38, 2014. doi:10.3389/fninf.2014.00038.

PYTHON MODULE INDEX

n

INDEX

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W