
propertyestimator Documentation

propertyestimator

Dec 03, 2019

USER GUIDE

- 1 Index 3**
- 1.1 Installing the Property Estimator 3
- 1.2 Getting Started 4
- 1.3 Physical Property Measurements 5
- 1.4 Physical Property Data Sets 8
- 1.5 API 10
- 1.6 Release History 253
- 1.7 Release Process 257

- Index 261**

The property estimator is a distributed framework from the [Open Forcefield Consortium](#) for storing, manipulating, and computing measured physical properties from simulation data.

Warning: This framework is still in **pre-alpha** and under heavy development. Although all steps have been taken to ensure correctness of the code and the results it produces, the authors accept no liability for any incorrectness any bugs or unintended behaviour may cause.

The framework currently has built in support for estimating the following properties:

Physical Property	Simulation		Reweighting	
	Implemented	Gradients	Implemented	Gradients
Density	Yes	Yes	Yes	Yes
Dielectric Constatant	Yes	Yes*	Yes	Yes*
$H_{\text{vaporization}}$	Yes	Yes	Yes	Yes*
H_{mixing}	Yes	Yes*	Yes	Yes*
V_{excess}	Yes	Yes*	Yes	Yes*
$G_{\text{solvation}}$	Yes*	No	No	No

* Entries marked with an asterisk are implemented but have not yet been extensively tested and validated.

User Guide

- *Installing the Property Estimator*
- *Getting Started*
- *Physical Property Measurements*
- *Physical Property Data Sets*

1.1 Installing the Property Estimator

The Property Estimator is currently installable both from source and through `conda`. Whichever route is chosen, it is recommended to install the estimator within a conda environment, and allow the conda package manager to install the required dependencies.

More information about conda and instructions to perform a lightweight miniconda installation [can be found here](#). It will be assumed that these have been followed and conda is available on your machine.

1.1.1 Installation from Conda

To install the `propertyestimator` from the `omnia` channel, simply run:

```
conda install -c openeye -c omnia propertyestimator
```

1.1.2 Optional Dependencies

To parameterize systems with the Amber `tLeap` tool using a `TLeapForceFieldSource` the `ambertools19` package must be installed:

```
conda install -c ambermd 'ambertools ==19.0'
```

1.1.3 Installation from Source

To install Property Estimator from source, clone the repository from [github](#):

```
git clone https://github.com/openforcefield/propertyestimator.git
cd propertyestimator
```

Create a custom conda environment which contains the required dependencies and activate it:

```
conda env create --name propertyestimator --file devtools/conda-envs/test_env.yaml
conda activate propertyestimator
```

The final step is to install the estimator itself:

```
python setup.py develop
```

And that's it!

1.2 Getting Started

The `propertyestimator` currently exists as two key components:

- a client object which the user can use to request the estimation of data sets of physical properties.
- a server object which accepts requests from a client and performs the estimations.

Warning: These instructions are still a work in progress, and may not run as expected.

1.2.1 Creating an Estimator Server

The `PropertyEstimatorServer` class creates objects that handle property estimation of all of the properties in a dataset given a set.

Create the file `run_server.py`. Tell server to log to file in case of failure:

```
setup_timestamp_logging()
```

Create directory structure to store intermediary results:

```
# Set the name of the directory in which all temporary files
# will be generated.
working_directory = 'working_directory'

# Remove any existing data.
if path.isdir(working_directory):
    shutil.rmtree(working_directory)
```

Set up a calculation backend. Different backends will take different optional arguments, but here is an example that will launch a backend with a single worker process:

```
# Create a calculation backend to perform workflow
# calculations on.
calculation_backend = DaskLocalCluster(1)
```

Set up storage the storage backend which will cache any generated simulation data:

```
# Create a backend to handle storing and retrieving
# cached simulation data.
storage_backend = LocalFileStorage()
```

Start the server running:


```
# Create a server instance.
property_server = server.PropertyEstimatorServer(calculation_backend,
                                                storage_backend,
                                                working_directory=working_directory)

# Tell the server to start listening for incoming
# estimation requests.
property_server.start_listening_loop()
```

To start the server, call the following command from the command line:

```
python run_server.py
```

The server will wait for requests until killed.

1.2.2 Submitting Estimation Requests

Create the file `run_client.py` Load in the data set of properties to estimate, and the force field parameters to use in the calculations:

```
# Load in the data set of interest.
data_set = ThermoMLDataSet.from_file(get_data_filename('properties/single_density.xml
↳'))

# Load in the force field to use.
force_field_source = SmirnoffForceFieldSource.from_path('smirnoff99Frosst-1.1.0.offxml
↳')
```

Create the client object and use it to send the estimation request to the server:

```
# Create the client object.
property_estimator = client.PropertyEstimatorClient()
# Submit the request to a running server.
result = property_estimator.request_estimate(data_set, force_field_source)
```

Query the result until all of the properties have either been estimated or have errored:

```
# Wait for the results synchronously.
results = request.results(True)
logging.info('The server has returned a response: {}'.format(result))
```

Save the results to a file:

```
with open('results.json', 'w') as file:

    json_results = json.dump(results, file, sort_keys=True, indent=2,
                             separators=(',', ': '), cls=TypedJSONEncoder)
```

1.3 Physical Property Measurements

Warning: This text is now out of date, but will be updated in future to reflect the latest version of the framework.

Physical property measurements are measured properties of a substance that provide some information about the physical parameters that define the interactions within the substance.

A physical property is defined by a combination of:

- A `Mixture` specifying the substance that the measurement was performed on
- A `ThermodynamicState` specifying the thermodynamic conditions under which the measurement was performed
- A `PhysicalProperty` is the physical property that was measured
- A `MeasurementMethod` specifying the kind of measurement that was performed

An example of each:

- `Mixture`: a 0.8 mole fraction mixture of ethanol and water
- `ThermodynamicState`: 298 kelvin, 1 atmosphere
- `PhysicalProperty`: mass density
- `MeasurementMethod`: vibrating tube method

1.3.1 Physical substances

We generally use the concept of a liquid or gas `Mixture`, which is a subclass of `Substance`.

A simple liquid has only one component:

```
liquid = Mixture()
liquid.add_component('water')
```

A binary mixture has two components:

```
binary_mixture = Mixture()
binary_mixture.add_component('water', mole_fraction=0.2)
binary_mixture.add_component('methanol') # assumed to be rest of mixture if no mole_
↳fraction specified
```

A ternary mixture has three components:

```
ternary_mixture = Mixture()
ternary_mixture.add_component('ethanol', mole_fraction=0.2)
ternary_mixture.add_component('methanol', mole_fraction=0.2)
ternary_mixture.add_component('water')
```

The infinite dilution of one solute within a solvent or mixture is also specified as a `Mixture`, where the solute has zero mole fraction:

```
infinite_dilution = Mixture()
infinite_dilution.add_component('phenol', impurity=True) # infinite dilution; one_
↳copy only of the impurity
infinite_dilution.add_component('water')
```

You can iterate over the components in a mixture:

```
for component in mixture.components:
    print (component.iupac_name, component.mole_fraction)
```

retrieve a component by name:

```
component = mixture.components['ethanol']
```

or get the number of components in a mixture:

```
ncomponents = mixture.ncomponents
```

or check if a component is an impurity:

```
if component.impurity == True:
    ...
```

1.3.2 Thermodynamic states

A `ThermodynamicState` specifies a combination of thermodynamic parameters (e.g. temperature, pressure) at which a measurement is performed.

```
from simtk import unit
thermodynamic_state = ThermodynamicState(pressure=500*unit.kilopascals,
↳temperature=298.15*unit.kelvin)
```

We use the `simtk.unit` unit system from [OpenMM](#) for units (though we may later migrate to `pint` for portability).

1.3.3 Physical property measurements

A `MeasuredPhysicalProperty` is a combination of `Substance`, `ThermodynamicState`, and a unit-bearing measured property value and uncertainty:

```
# Define mixture
mixture = Mixture()
mixture.addComponent('water', mole_fraction=0.2)
mixture.addComponent('methanol')

# Define thermodynamic state
thermodynamic_state = ThermodynamicState(pressure=500*unit.kilopascals,
↳temperature=298.15*unit.kelvin)

# Define measurement
measurement = ExcessMolarEnthalpy(substance, thermodynamic_state, value=83.
↳3863244*unit.kilojoules_per_mole,
                                uncertainty=0.1220794866*unit.kilojoules_per_mole)
```

The various properties are all subclasses of `MeasuredPhysicalProperty` and generally follow the `<ePropName/> ThermoML` tag names.

Some examples of `MeasuredPhysicalProperty`:

- `MassDensity` - mass density
- `ExcessMolarEnthalpy` - excess partial apparent molar enthalpy
- `HeatCapacity` - molar heat capacity at constant pressure

A [roadmap of physical properties to be implemented](#) is available.

Please raise an issue if your physical property of interest is not listed!

Each `MeasuredPhysicalProperty` has several properties:

- `.substance` - the `Mixture` for which the measurement was made
- `.thermodynamic_state` - the `ThermodynamicState` at which the measurement was made
- `.measurement_method` - the `MeasurementMethod` used to measure the physical property
- `.value` - the unit-bearing measurement value
- `.uncertainty` - the standard uncertainty of the measurement
- `.reference` - the literature reference (if available) for the measurement
- `.DOI` - the literature reference DOI (if available) for the measurement

The value, uncertainty, reference, and DOI do not necessarily need to be defined for a dataset in order for property calculations to be performed.

1.4 Physical Property Data Sets

Warning: This text is now out of date, but will be updated in future to reflect the latest version of the framework.

A `PhysicalPropertyDataset` is a collection of `MeasuredPhysicalProperty` objects that are related in some way.

```
dataset = PhysicalPropertyDataset([measurement1, measurement2])
```

The dataset is iterable:

```
dataset = PhysicalPropertyDataset([measurement1, measurement2])

for measurement in dataset:
    print measurement.value
```

and has accessors to retrieve DOIs and references associated with measurements in the dataset:

```
# Print the DOIs associated with this dataset
print(dataset.DOIs)

# Print the references associated with this dataset
print(dataset.references)
```

For convenience, you can retrieve the dataset as a `pandas DataFrame`:

```
dataset.to_pandas()
```

1.4.1 ThermoML datasets

A `ThermoMLDataset` object represents a physical property dataset stored in the IUPAC-standard `ThermoML` for specifying thermodynamic properties in XML format. `ThermoMLDataset` is a subclass of `PhysicalPropertyDataset`, and provides the same API interface (in addition to some `ThermoML`-specific methods).

Direct access to the [NIST ThermoML Archive](#) is supported for obtaining physical property measurements in this format directly from the NIST TRC repository.

For example, to retrieve the [ThermoML dataset](#) that accompanies [this paper](#), we can simply use the DOI `10.1016/j.jct.2005.03.012` as a key for creating a `PhysicalPropertyDataset` subclassed object from the ThermoML Archive:

```
dataset = ThermoMLDataset(doi='10.1016/j.jct.2005.03.012')
```

You can also specify multiple ThermoML Archive keys to create a dataset from multiple ThermoML files:

```
thermoml_keys = ['10.1021/acs.jced.5b00365', '10.1021/acs.jced.5b00474']
dataset = ThermoMLDataset(doi=thermoml_keys)
```

It is also possible to specify ThermoML datasets housed at other locations, such as

```
dataset = ThermoMLDataset(url='http://openforcefieldgroup.org/thermoml-datasets')
```

or

```
dataset = ThermoMLDataset(url='file:///Users/choderaj/thermoml')
```

or

```
dataset = ThermoMLDataset(doi=['10.1021/acs.jced.5b00365', '10.1021/acs.jced.5b00474',
↪ ],
                           url='http://openforcefieldgroup.org/thermoml-datasets')
```

or from ThermoML and a different URL:

```
dataset = ThermoMLDataset(doi=thermoml_keys)
dataset.retrieve(doi=local_keys, url='http://openforcefieldgroup.org/thermoml-datasets',
↪ )
```

You can see which DOIs contribute to the current `ThermoMLDataset` with the convenience functions:

```
print(dataset.DOIs)
```

NIST has compiled a JSON frame of corrections to uncertainties.

These can be used to update or correct data uncertainties and discard outliers using `applyNISTUncertainties()`:

```
# Modify uncertainties according to NIST evaluation
dataset.apply_nist_uncertainties(nist_uncertainties, adjust_uncertainties=True,
↪ discard_outliers=True)
```

Todo:

- We should merge any other useful parts of the [ThermoPyL API](#) in here.
-

1.4.2 Other datasets

In future, we will add interfaces to other online datasets, such as

- [BindingDB](#) for retrieving [host-guest binding affinity](#) datasets.

Developer Documentation

- [API](#)
- [Release History](#)
- [Release Process](#)

1.5 API

A set of API documents for this projects classes and modules.

1.5.1 Client Side API

<i>PropertyEstimatorClient</i>	The PropertyEstimatorClient is the main object that users of the property estimator will interface with.
<i>PropertyEstimatorOptions</i>	Represents the options options that can be passed to the property estimation server backend.
<i>PropertyEstimatorSubmission</i>	Represents a set of properties to be estimated by the server backend, the parameters which will be used to estimate them, and options about how the properties will be estimated.
<i>PropertyEstimatorResult</i>	Represents the results of attempting to estimate a set of physical properties using the property estimator server backend.
<i>ConnectionOptions</i>	The set of options to use when connecting to a <i>PropertyEstimatorServer</i>

PropertyEstimatorClient

class `propertyestimator.client.PropertyEstimatorClient` (*connection_options=<propertyestimator.client.ConnectionOptions object>*)

The PropertyEstimatorClient is the main object that users of the property estimator will interface with. It is responsible for requesting that a PropertyEstimatorServer estimates a set of physical properties, as well as querying for when those properties have been estimated.

The PropertyEstimatorClient supports two main workflows: one where a PropertyEstimatorServer lives on a remote supercomputing cluster where all of the expensive calculations will be run, and one where the users local machine acts as both the server and the client, and all calculations will be performed locally.

Warning: While the API of this class is now close to being final, the internals and implementation are still heavily under development and is subject to rapid changes.

Examples

Setting up the client instance:

```
>>> from propertyestimator.client import PropertyEstimatorClient
>>> property_estimator = PropertyEstimatorClient()
```

If the PropertyEstimatorServer is not running on the local machine, you will need to specify its address and the port that it is listening on:

```
>>> from propertyestimator.client import ConnectionOptions
>>>
>>> connection_options = ConnectionOptions(server_address='server_address',
>>>                                       server_port=8000)
>>> property_estimator = PropertyEstimatorClient(connection_options)
```

To asynchronously submit a request to the running server using the default estimator options:

```
>>> # Load in the data set of properties which will be used for comparisons
>>> from propertyestimator.datasets import ThermoMLDataSet
>>> data_set = ThermoMLDataSet.from_doi('10.1016/j.jct.2016.10.001')
>>> # Filter the dataset to only include densities measured between 130-260 K
>>> from propertyestimator.properties import Density
>>>
>>> data_set.filter_by_property_types(Density)
>>> data_set.filter_by_temperature(min_temperature=130*unit.kelvin, max_
↳temperature=260*unit.kelvin)
>>>
>>> # Load in the force field parameters
>>> from openforcefield.typing.engines import smirnoff
>>> from propertyestimator.forcefield import SmirnoffForceFieldSource
>>> smirnoff_force_field = smirnoff.ForceField('smirnoff99Frosst-1.1.0.offxml')
>>> force_field_source = SmirnoffForceFieldSource.from_object(smirnoff_force_
↳field)
>>>
>>> request = property_estimator.request_estimate(data_set, force_field_source)
```

The status of the request can be asynchronously queried by calling

```
>>> results = request.results()
```

or the main thread can be blocked until the results are available by calling

```
>>> results = request.results(synchronous=True)
```

How the property set will be estimated can easily be controlled by passing a PropertyEstimatorOptions object to the estimate commands.

The calculations layers which will be used to estimate the properties can be controlled for example like so:

```
>>> from propertyestimator.layers import ReweightingLayer, SimulationLayer
>>>
>>> options = PropertyEstimatorOptions(allowed_calculation_layers = [
↳[ReweightingLayer,
↳SimulationLayer])
>>>
>>> request = property_estimator.request_estimate(data_set, force_field_source,
↳options)
```

Options for how properties should be estimated can be set on a per property, and per layer basis. For example, the relative uncertainty that properties should be estimated to within by the SimulationLayer can be set as:

```
>>> from propertyestimator.workflow import WorkflowOptions
>>>
>>> workflow_options = WorkflowOptions(WorkflowOptions.ConvergenceMode,
↳RelativeUncertainty,
```

(continues on next page)

(continued from previous page)

```

>>>                                     relative_uncertainty_fraction=0.1)
>>> options.workflow_options = {
>>>     'Density': {'SimulationLayer': workflow_options},
>>>     'Dielectric': {'SimulationLayer': workflow_options}
>>> }

```

Or alternatively, as absolute uncertainty tolerance can be set as:

```

>>> density_options = WorkflowOptions(WorkflowOptions.ConvergenceMode.
↳AbsoluteUncertainty,
>>>                                     absolute_uncertainty=0.0002 * unit.gram /_
↳unit.milliliter)
>>> dielectric_options = WorkflowOptions(WorkflowOptions.ConvergenceMode.
↳AbsoluteUncertainty,
>>>                                     absolute_uncertainty=0.02 * unit.
↳dimensionless)
>>>
>>> options.workflow_options = {
>>>     'Density': {'SimulationLayer': density_options},
>>>     'Dielectric': {'SimulationLayer': dielectric_options}
>>> }

```

The gradients of the observables of interest with respect to a number of chosen parameters can be requested by passing a *parameter_gradient_keys* parameter. In the below example, gradients will be calculated with respect to both the bond length parameter for the [#6:1]-[#8:2] chemical environment, and the bond angle parameter for the [:1]-[#8:2]-[:3] chemical environment:

```

>>> from propertyestimator.properties import ParameterGradientKey
>>>
>>> parameter_gradient_keys = [
>>>     ParameterGradientKey('Bonds', '#6:1-#8:2', 'length')
>>>     ParameterGradientKey('Angles', '*:1-#8:2-[:3]', 'angle')
>>> ]
>>>
>>> request = property_estimator.request_estimate(data_set, force_field_source,
↳options, parameter_gradient_keys)
>>>

```

__init__ (*connection_options*=<propertyestimator.client.ConnectionOptions object>)

Constructs a new PropertyEstimatorClient object.

Parameters *connection_options* (*ConnectionOptions*) – The options used when connecting to the calculation server.

Methods

<code>__init__</code> ([<i>connection_options</i>])	Constructs a new PropertyEstimatorClient object.
<code>request_estimate</code> (<i>property_set</i> , ...[, ...])	Requests that a PropertyEstimatorServer attempt to estimate the provided property set using the supplied force field and estimator options.

Attributes

server_address

server_port

class Request (*request_id, connection_options, client=None*)

An object representation of a estimation request which has been sent to a *PropertyEstimatorServer* instance. This object can be used to query and retrieve the results of the request, or be stored to retrieve the request at some point in the future.

property id

The id of the submitted request.

Type *str*

property server_address

The address of the server that the request was sent to.

Type *str*

property server_port

The port that the server is listening on.

json()

Returns a JSON representation of the *Request* object.

Returns The JSON representation of the *Request* object.

Return type *str*

classmethod from_json (*json_string*)

Creates a new *Request* object from a JSON representation.

Parameters *json_string* (*str*) – The JSON representation of the *Request* object.

Returns The created *Request* object.

Return type *str*

results (*synchronous=False, polling_interval=5*)

Retrieve the results of an estimate request.

Parameters

- **synchronous** (*bool*) – If true, this method will block the main thread until the server either returns a result or an error.
- **polling_interval** (*int*) – If running synchronously, this is the time interval (seconds) between checking if the calculation has finished.

Returns

Returns either the results of the requested estimate, or any exceptions which were raised.

If the method is run synchronously then this method will block the main thread until all of the requested properties have been estimated, or an exception is returned.

Return type *PropertyEstimatorResult* or *PropertyEstimatorException*

request_estimate (*property_set, force_field_source, options=None, parameter_gradient_keys=None*)

Requests that a *PropertyEstimatorServer* attempt to estimate the provided property set using the supplied force field and estimator options.

Parameters

- **property_set** (*PhysicalPropertyDataSet*) – The set of properties to attempt to estimate.
- **force_field_source** (*ForceFieldSource* or *openforcefield.typing.engines.smirnoff.ForceField*) – The source of the force field parameters to use for the calculations.

- **options** (`PropertyEstimatorOptions`, *optional*) – A set of estimator options. If None, default options will be used.
- **parameter_gradient_keys** (*list of ParameterGradientKey, optional*) – A list of references to all of the parameters which all observables should be differentiated with respect to.

Returns An object which will provide access the the results of the request.

Return type `PropertyEstimatorClient.Request`

PropertyEstimatorOptions

class `propertyestimator.client.PropertyEstimatorOptions` (*allowed_calculation_layers=None, allow_protocol_merging=True*)

Represents the options options that can be passed to the property estimation server backend.

Warning:

- This class is still heavily under development and is subject to rapid changes.

allowed_calculation_layers

A list of allowed calculation layers. The order of the layers in the list is the order that the calculator will attempt to execute the layers in.

Type list of str or list of class

workflow_schemas

A dictionary of the WorkflowSchema which will be used to calculate any properties. The dictionary key represents the type of property the schema will calculate. The dictionary will be automatically populated with defaults if no entries are added.

Type dict of str and dict of str and WorkflowSchema

workflow_options

The set of options which will be used when setting up the default estimation workflows, where the string key here is the property for which the options apply. As an example, the target (relative or absolute) uncertainty of each property may be set using these options.

If None, a set of defaults will be applied when the properties are sent to a server for estimation. The current set of defaults will ensure that properties are estimated with an uncertainty which is less than or equal to the experimental uncertainty of a property.

Type dict of str and dict of str and WorkflowOptions, optional

allow_protocol_merging

If true, allows individual identical steps in a property estimation workflow to be merged.

Type bool, default = True

__init__ (*allowed_calculation_layers=None, allow_protocol_merging=True*)

Constructs a new PropertyEstimatorOptions object.

Parameters

- **allowed_calculation_layers** (*list of str or list of class*) – A list of allowed calculation layers. The order of the layers in the list is the order that the calculator will attempt to execute the layers in.

If None, all registered calculation layers are set as allowed.

- **allow_protocol_merging** (*bool*, *default = True*) – If true, allows individual identical steps in a property estimation workflow to be merged.

Methods

<code>__init__</code> ([allowed_calculation_layers, ...])	Constructs a new PropertyEstimatorOptions object.
<code>json</code> ()	Creates a JSON representation of this class.
<code>parse_json</code> (string_contents[, encoding])	Parses a typed json string into the corresponding class structure.

`json`()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

`classmethod parse_json` (*string_contents*, *encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

PropertyEstimatorSubmission

```
class propertyestimator.client.PropertyEstimatorSubmission (properties=None,
                                                         force_field_source=None,
                                                         options=None,
                                                         parameter_gradient_keys=None)
```

Represents a set of properties to be estimated by the server backend, the parameters which will be used to estimate them, and options about how the properties will be estimated.

Warning: This class is still heavily under development and is subject to rapid changes.

properties

The list of physical properties to estimate.

Type list of PhysicalProperty

options

The options which control how the *properties* are estimated.

Type *PropertyEstimatorOptions*

force_field_source

The source of the force field parameters used during the calculations.

Type *ForceFieldSource*

`__init__` (*properties=None, force_field_source=None, options=None, parameter_gradient_keys=None*)

Constructs a new PropertyEstimatorSubmission object.

Parameters

- **properties** (*list of PhysicalProperty*) – The list of physical properties to estimate.
- **options** (*PropertyEstimatorOptions*) – The options which control how the *properties* are estimated.
- **force_field_source** (*ForceFieldSource*) – The source of the force field parameters used during the calculations.
- **parameter_gradient_keys** (*list of ParameterGradientKey*) – A list of references to all of the parameters which all observables should be differentiated with respect to.

Methods

<code>__init__</code> ([<i>properties, force_field_source, ...</i>])	Constructs a new PropertyEstimatorSubmission object.
<code>json</code> ()	Creates a JSON representation of this class.
<code>parse_json</code> (<i>string_contents</i> [, <i>encoding</i>])	Parses a typed json string into the corresponding class structure.

`json` ()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type *str*

classmethod `parse_json` (*string_contents, encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type *Any*

PropertyEstimatorResult

class `propertyestimator.client.PropertyEstimatorResult` (*result_id=""*)

Represents the results of attempting to estimate a set of physical properties using the property estimator server backend.

Warning: This class is still heavily under development and is subject to rapid changes.

id

The unique id assigned to this result set by the server.

Type `str`

queued_properties

A dictionary of the properties which have yet to be estimated by the server.

Type dict of str and PhysicalProperty

estimated_properties

A dictionary of the properties which were successfully estimated, where the dictionary key is the unique id of the property being estimated.

Type dict of str and PhysicalProperty

unsuccessful_properties

A dictionary of the properties which could not be estimated by the server.

Type dict of str and PhysicalProperty

exceptions

A list of the exceptions that were raised when unsuccessfully carrying out this estimation request.

Type list of PropertyEstimatorException

__init__ (*result_id*=")

Constructs a new PropertyEstimatorResult object.

Parameters **result_id** (*str*) – The unique id assigned to this result set by the server.

Methods

<code>__init__</code> ([<i>result_id</i>])	Constructs a new PropertyEstimatorResult object.
<code>json</code> ()	Creates a JSON representation of this class.
<code>parse_json</code> (<i>string_contents</i> [, <i>encoding</i>])	Parses a typed json string into the corresponding class structure.

json ()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

classmethod `parse_json` (*string_contents*, *encoding*='utf8')

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str* or *bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

ConnectionOptions

class propertyestimator.client.**ConnectionOptions** (*server_address='localhost',
server_port=8000*)

The set of options to use when connecting to a *PropertyEstimatorServer*

server_address

The address of the server to connect to.

Type *str*

server_port

The port number that the server is listening on.

Type *int*

Warning: This class is still heavily under development and is subject to rapid changes.

__init__ (*server_address='localhost', server_port=8000*)

Constructs a new ConnectionOptions object.

Parameters

- **server_address** (*str*) – The address of the server to connect to.
- **server_port** (*int*) – The port number that the server is listening on.

Methods

<code>__init__([server_address, server_port])</code>	Constructs a new ConnectionOptions object.
<code>json()</code>	Creates a JSON representation of this class.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.

Attributes

<code>server_address</code>
<code>server_port</code>

json ()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type *str*

classmethod **parse_json** (*string_contents, encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

Force Field Sources

<i>ForceFieldSource</i>	A helper object to define the source of a force field and any associated meta data, such as version, file paths, or generation options.
<i>SmirnoffForceFieldSource</i>	A wrapper around force fields based on the SMIRks Native Open Force Field (SMIRNOFF) specification.
<i>TLeapForceFieldSource</i>	A wrapper around Amber force fields which may be applied via the <i>tleap</i> software package.
<i>LigParGenForceFieldSource</i>	A wrapper and the OPLSAAM force field which can be applied via the LigParGen server .

ForceFieldSource

class `propertyestimator.forcefield.ForceFieldSource`

A helper object to define the source of a force field and any associated meta data, such as version, file paths, or generation options.

Notes

It is likely that this class and classes based off of it will not be permanent fixtures of the framework, but rather will exist until the force fields can be stored in a uniform format / object model.

`__init__()`

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__</code>	Initialize self.
<code>json()</code>	Creates a JSON representation of this class.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.

`json()`

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

classmethod `parse_json(string_contents, encoding='utf8')`

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (`str` or `bytes`) – The typed json string.
- **encoding** (`str`) – The encoding of the `string_contents`.

Returns The parsed class.

Return type Any

SmirnoffForceFieldSource

class `propertyestimator.forcefield.SmirnoffForceFieldSource` (*inner_xml=None*)

A wrapper around force fields based on the SMIRks Native Open Force Field (SMIRNOFF) specification.

`__init__` (*inner_xml=None*)

Constructs a new `SmirnoffForceFieldSource` object

Parameters `inner_xml` (*str, optional*) – A string containing the xml representation of the force field.

Methods

<code>__init__</code> (<i>inner_xml</i>)	Constructs a new <code>SmirnoffForceFieldSource</code> object
<code>from_object</code> (<i>force_field</i>)	Creates a new <code>SmirnoffForceFieldSource</code> from an existing <code>ForceField</code> object
<code>from_path</code> (<i>file_path</i>)	Creates a new <code>SmirnoffForceFieldSource</code> from the file path to a <code>ForceField</code> object.
<code>json</code> ()	Creates a JSON representation of this class.
<code>parse_json</code> (<i>string_contents</i> [, <i>encoding</i>])	Parses a typed json string into the corresponding class structure.
<code>to_force_field</code> ()	Returns the SMIRNOFF force field created from this source.

`to_force_field`()

Returns the SMIRNOFF force field created from this source.

Returns The created force field.

Return type `openforcefield.typing.engines.smirnoff.ForceField`

classmethod `from_object` (*force_field*)

Creates a new `SmirnoffForceFieldSource` from an existing `ForceField` object

Notes

All cosmetic attributes will be discarded.

Parameters `force_field` (`openforcefield.typing.engines.smirnoff.ForceField`) – The existing force field.

Returns The created object.

Return type `SmirnoffForceFieldSource`

classmethod `from_path` (*file_path*)

Creates a new `SmirnoffForceFieldSource` from the file path to a `ForceField` object.

Notes

All cosmetic attributes will be discarded.

Parameters `file_path` (*str*) – The file path to the force field object. This may also be the name of a file which can be loaded via an entry point.

Returns The created object.

Return type *SmirnoffForceFieldSource*

json()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type *str*

classmethod parse_json (*string_contents*, *encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str* or *bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

TLeapForceFieldSource

```
class propertyestimator.forcefield.TLeapForceFieldSource (leap_source='leaprc.gaff2',
                                                           cutoff=<Quantity(9.0,
                                                           'angstrom')>)
```

A wrapper around Amber force fields which may be applied via the *tLeap* software package.

Notes

Currently this only supports force fields which are installed alongside *tLeap*.

```
__init__ (leap_source='leaprc.gaff2', cutoff=<Quantity(9.0, 'angstrom')>)
```

Constructs a new TLeapForceFieldSource object

Parameters

- **leap_source** (*str*) – The parameter file which should be sourced by *leap* when applying the force field. Currently only *'leaprc.gaff'* and *'leaprc.gaff2'* are supported.
- **cutoff** (*unit.Quantity*) – The non-bonded interaction cutoff.

Examples

To create a source for the GAFF force field with tip3p water:

```
>>> amber_gaff_source = TLeapForceFieldSource('leaprc.gaff')
```

To create a source for the GAFF 2 force field with tip3p water:

```
>>> amber_gaff_2_source = TLeapForceFieldSource('leaprc.gaff2')
```

Methods

```
__init__ ([leap_source, cutoff])
```

Constructs a new TLeapForceFieldSource object

Continued on next page

Table 12 – continued from previous page

<code>json()</code>	Creates a JSON representation of this class.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.

Attributes

<code>cutoff</code>	The non-bonded interaction cutoff.
<code>leap_source</code>	The parameter file which should be sourced by <code>leap</code> when applying the force field.

property leap_source

The parameter file which should be sourced by `leap` when applying the force field.

Type list of str

property cutoff

The non-bonded interaction cutoff.

Type unit.Quantity

json()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents, encoding='utf8')

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str* or *bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

LigParGenForceFieldSource

```
class propertyestimator.forcefield.LigParGenForceFieldSource (preferred_charge_model=<ChargeModel.C
    '1.14*CMIA-
    LBCC'>, cutoff=<Quantity(9.0,
    'angstrom')>,
    request_url="",
    download_url="")
```

A wrapper and the OPLSAAM force field which can be applied via the [LigParGen server](#).

References

- [1] **Potential energy functions for atomic-level simulations of water and organic and biomolecular systems.** Jorgensen, W. L.; Tirado-Rives, J. Proc. Nat. Acad. Sci. USA 2005, 102, 6665-6670

- [2] **1.14*CM1A-LBCC: Localized Bond-Charge Corrected CM1A Charges for Condensed-Phase Simulations.** Dodda, L. S.; Vilseck, J. Z.; Tirado-Rives, J.; Jorgensen, W. L. J. Phys. Chem. B, 2017, 121 (15), pp 3864-3870
- [3] **LigParGen web server: An automatic OPLS-AA parameter generator for organic ligands.** Dodda, L. S.; Cabeza de Vaca, I.; Tirado-Rives, J.; Jorgensen, W. L. Nucleic Acids Research, Volume 45, Issue W1, 3 July 2017, Pages W331-W336

`__init__` (*preferred_charge_model*=<*ChargeModel.CM1A_1_14_LBCC*: '1.14*CM1A-LBCC'>, *cutoff*=<*Quantity*(9.0, 'angstrom')>, *request_url*="", *download_url*="")
Constructs a new LigParGenForceFieldSource object

Parameters

- **preferred_charge_model** (*ChargeModel*) – The preferred charge model to apply. In some cases the preferred charge model may not be applicable (e.g. 1.14*CM1A-LBCC may only be applied to neutral molecules) and so another model may be applied in its place.
- **cutoff** (*unit.Quantity*) – The non-bonded interaction cutoff.
- **request_url** (*str*) – The URL of the LIGPARGEN server file to send the parametrization to request to.
- **download_url** (*str*) – The URL of the LIGPARGEN server file to download the results of a request from.

Methods

<code>__init__</code> (<i>preferred_charge_model</i> , <i>cutoff</i> , ...)	Constructs a new LigParGenForceFieldSource object
<code>json</code> ()	Creates a JSON representation of this class.
<code>parse_json</code> (<i>string_contents</i> [, <i>encoding</i>])	Parses a typed json string into the corresponding class structure.

Attributes

<code>cutoff</code>	The non-bonded interaction cutoff.
<code>download_url</code>	The URL of the LIGPARGEN server file to download the results of a request from.
<code>preferred_charge_model</code>	The preferred charge model to apply.
<code>request_url</code>	The URL of the LIGPARGEN server file to send the parametrization to request to.

class ChargeModel

An enumeration.

property preferred_charge_model

The preferred charge model to apply. In some cases the preferred charge model may not be applicable (e.g. 1.14*CM1A-LBCC may only be applied to neutral molecules) and so another model may be applied in its place.

Type *ChargeModel*

property cutoff

The non-bonded interaction cutoff.

Type `unit.Quantity`

property request_url

The URL of the LIGPARGEN server file to send the parametrization to request to.

Type `str`

property download_url

The URL of the LIGPARGEN server file to download the results of a request from.

Type `str`

json()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

classmethod parse_json (*string_contents*, *encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type `Any`

Gradient Estimation

ParameterGradientKey

ParameterGradient

ParameterGradientKey

class `propertyestimator.properties.ParameterGradientKey` (*tag=None*, *smirks=None*, *attribute=None*)

__init__ (*tag=None*, *smirks=None*, *attribute=None*)

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__</code> ([<i>tag</i> , <i>smirks</i> , <i>attribute</i>])	Initialize self.
---	------------------

Attributes

<code>attribute</code>
<code>smirks</code>
<code>tag</code>

ParameterGradient

class propertyestimator.properties.**ParameterGradient** (*key=None, value=None*)

`__init__` (*key=None, value=None*)
Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__</code> ([key, value])	Initialize self.
--------------------------------------	------------------

Attributes

key
value

1.5.2 Server Side API

PropertyEstimatorServer

The object responsible for coordinating all properties estimations to to be ran using the property estimator, in addition to deciding at which fidelity a property will be calculated.

PropertyEstimatorServer

class propertyestimator.server.**PropertyEstimatorServer** (*calculation_backend, storage_backend, port=8000, working_directory='working-data'*)

The object responsible for coordinating all properties estimations to to be ran using the property estimator, in addition to deciding at which fidelity a property will be calculated.

It acts as a server, which receives submitted jobs from clients launched via the property estimator.

Warning: This class is still heavily under development and is subject to rapid changes.

Notes

Methods to handle the TCP messages are based on the StackOverflow response from A. Jesse Jiryu Davis: <https://stackoverflow.com/a/40257248>

Examples

Setting up a general server instance using a dask LocalCluster backend:

```

>>> # Create the backend which will be responsible for distributing the_
↳calculations
>>> from propertyestimator.backends import DaskLocalCluster, ComputeResources
>>> calculation_backend = DaskLocalCluster(1)
>>>
>>> # Calculate the backend which will be responsible for storing and retrieving
>>> # the data from previous calculations
>>> from propertyestimator.storage import LocalFileStorage
>>> storage_backend = LocalFileStorage()
>>>
>>> # Create the server to which all estimation requests will be submitted
>>> from propertyestimator.server import PropertyEstimatorServer
>>> property_server = PropertyEstimatorServer(calculation_backend, storage_
↳backend)
>>>
>>> # Instruct the server to listen for incoming requests
>>> property_server.start_listening_loop()

```

`__init__(calculation_backend, storage_backend, port=8000, working_directory='working-data')`
 Constructs a new PropertyEstimatorServer object.

Parameters

- **calculation_backend** (PropertyEstimatorBackend) – The backend to use for executing calculations.
- **storage_backend** (PropertyEstimatorStorage) – The backend to use for storing information from any calculations.
- **port** (*int*) – The port on which to listen for incoming client requests.
- **working_directory** (*str*) – The local directory in which to store all local, temporary calculation data.

Methods

<code>__init__(calculation_backend, storage_backend)</code>	Constructs a new PropertyEstimatorServer object.
<code>add_socket(socket)</code>	Singular version of <i>add_sockets</i> .
<code>add_sockets(sockets)</code>	Makes this server start accepting connections on the given sockets.
<code>bind(port[, address, family, backlog, ...])</code>	Binds this server to the given port on the given address.
<code>handle_stream(stream, address)</code>	A routine to handle incoming requests from a property estimator TCP client.
<code>listen(port[, address])</code>	Starts accepting connections on the given port.
<code>start([num_processes])</code>	Starts this server in the <i>.IOLoop</i> .
<code>start_listening_loop()</code>	Starts the main (blocking) server IOLoop which will run until the user kills the process.
<code>stop()</code>	Stops the property calculation server and its provided backend.

class ServerEstimationRequest (*estimation_id="", queued_properties=None, options=None, force_field_id=None, parameter_gradient_keys=None*)

Represents a request for the server to estimate a set of properties. Such requests are expected to only estimate properties for a single system (e.g. fixed components in a fixed ratio)

json()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type *str*

classmethod parse_json (*string_contents*, *encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str* or *bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

async handle_stream (*stream*, *address*)

A routine to handle incoming requests from a property estimator TCP client.

Notes

This method is based on the StackOverflow response from A. Jesse Jiryu Davis: <https://stackoverflow.com/a/40257248>

Parameters

- **stream** (*IOStream*) – An IO stream used to pass messages between the server and client.
- **address** (*str*) – The address from which the request came.

start_listening_loop()

Starts the main (blocking) server IOLoop which will run until the user kills the process.

stop()

Stops the property calculation server and it's provided backend.

add_socket (*socket*)

Singular version of *add_sockets*. Takes a single socket object.

add_sockets (*sockets*)

Makes this server start accepting connections on the given sockets.

The *sockets* parameter is a list of socket objects such as those returned by *~tornado.netutil.bind_sockets*. *add_sockets* is typically used in combination with that method and *tornado.process.fork_processes* to provide greater control over the initialization of a multi-process server.

bind (*port*, *address=None*, *family=<AddressFamily.AF_UNSPEC: 0>*, *backlog=128*, *reuse_port=False*)

Binds this server to the given port on the given address.

To start the server, call *start*. If you want to run this server in a single process, you can call *listen* as a shortcut to the sequence of *bind* and *start* calls.

Address may be either an IP address or hostname. If it's a hostname, the server will listen on all IP addresses associated with the name. Address may be an empty string or None to listen on all available interfaces. Family may be set to either *socket.AF_INET* or *socket.AF_INET6* to restrict to IPv4 or IPv6 addresses, otherwise both will be used if available.

The *backlog* argument has the same meaning as for *socket.listen <socket.socket.listen>*. The *reuse_port* argument has the same meaning as for *.bind_sockets*.

This method may be called multiple times prior to *start* to listen on multiple ports or interfaces.

Changed in version 4.4: Added the `reuse_port` argument.

listen (*port*, *address=""*)

Starts accepting connections on the given port.

This method may be called more than once to listen on multiple ports. *listen* takes effect immediately; it is not necessary to call *TCPServer.start* afterwards. It is, however, necessary to start the *.IOLoop*.

start (*num_processes=1*)

Starts this server in the *.IOLoop*.

By default, we run the server in this process and do not fork any additional child process.

If *num_processes* is *None* or ≤ 0 , we detect the number of cores available on this machine and fork that number of child processes. If *num_processes* is given and > 1 , we fork that specific number of sub-processes.

Since we use processes and not threads, there is no shared memory between any server code.

Note that multiple processes are not compatible with the autoreload module (or the `autoreload=True` option to *tornado.web.Application* which defaults to `True` when `debug=True`). When using multiple processes, no *IOLoops* can be created or referenced until after the call to *TCPServer.start(n)*.

1.5.3 Physical Property API

<i>PhysicalProperty</i>	Represents the value of any physical property and it's uncertainty.
<i>PropertyPhase</i>	An enum describing the phase a property was collected in.
<i>Source</i>	Container class for information about how a property was measured / calculated.
<i>MeasurementSource</i>	Contains any metadata about how a physical property was measured by experiment.
<i>CalculationSource</i>	Contains any metadata about how a physical property was calculated.

PhysicalProperty

```
class propertyestimator.properties.PhysicalProperty (thermodynamic_state=None,
                                                    phase=, substance=None,
                                                    value=None, uncertainty=None,
                                                    gradients=None, source=None)
```

Represents the value of any physical property and it's uncertainty.

It additionally stores the thermodynamic state at which the property was collected, the phase it was collected in, information about the composition of the observed system, and metadata about how the property was collected.

```
__init__ (thermodynamic_state=None, phase=, substance=None, value=None, uncertainty=None,
          gradients=None, source=None)
```

Constructs a new *PhysicalProperty* object.

Parameters

- **thermodynamic_state** (*ThermodynamicState*) – The thermodynamic state that the property was measured in.
- **phase** (*PropertyPhase*) – The phase that the property was measured in.

- **substance** (*Substance*) – The composition of the substance that was measured.
- **value** (*unit.Quantity*) – The value of the measured physical property.
- **uncertainty** (*unit.Quantity*) – The uncertainty in the measured value.
- **source** (*Source*) – The source of this property.

Methods

<code>__init__</code> ([thermodynamic_state, phase, ...])	Constructs a new <code>PhysicalProperty</code> object.
<code>get_default_workflow_schema</code> (<i>calculation_layer</i>)	Returns the default workflow schema to use for a specific calculation layer.
<code>json</code> ()	Creates a JSON representation of this class.
<code>parse_json</code> (<i>string_contents</i> [, <i>encoding</i>])	Parses a typed json string into the corresponding class structure.
<code>set_value</code> (<i>value</i> , <i>uncertainty</i>)	Set the value and uncertainty of this property.

Attributes

<code>metadata</code>	Additional metadata associated with this property, such as file paths to coordinate files or ...
<code>pressure</code>	The pressure at which the property was collected.
<code>temperature</code>	The temperature at which the property was collected.

property temperature

The temperature at which the property was collected.

Type `propertyestimator.unit.Quantity` or `None`

property pressure

The pressure at which the property was collected.

Type `propertyestimator.unit.Quantity` or `None`

property metadata

Additional metadata associated with this property, such as file paths to coordinate files or ...

All property metadata will be made accessible to property estimation workflows.

Type `dict` of `str` and `Any`

set_value (*value*, *uncertainty*)

Set the value and uncertainty of this property.

Parameters

- **value** (*propertyestimator.unit.Quantity*) – The value of the property.
- **uncertainty** (*propertyestimator.unit.Quantity*) – The uncertainty in the properties value.

static get_default_workflow_schema (*calculation_layer*, *options=None*)

Returns the default workflow schema to use for a specific calculation layer.

Parameters

- **calculation_layer** (*str*) – The calculation layer which will attempt to execute the workflow defined by this schema.

- **options** (`WorkflowOptions`) – The options to use when setting up the default workflows.

Returns The default workflow schema.

Return type `WorkflowSchema`

json()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

classmethod parse_json (`string_contents`, `encoding='utf8'`)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (`str` or `bytes`) – The typed json string.
- **encoding** (`str`) – The encoding of the `string_contents`.

Returns The parsed class.

Return type `Any`

PropertyPhase

class `propertyestimator.properties.PropertyPhase`

An enum describing the phase a property was collected in.

__init__()

Initialize self. See `help(type(self))` for accurate signature.

Attributes

Gas

Liquid

Solid

Undefined

Source

class `propertyestimator.properties.Source`

Container class for information about how a property was measured / calculated.

Todo: Swap this out with a more general provenance class.

__init__()

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__</code>	Initialize self.
<code>json()</code>	Creates a JSON representation of this class.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.

`json()`

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

classmethod `parse_json(string_contents, encoding='utf8')`

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

MeasurementSource

class `propertyestimator.properties.MeasurementSource(doi="", reference="")`

Contains any metadata about how a physical property was measured by experiment.

This class contains either the DOI and/or the reference, but must contain at least one as the observable must have a source, even if it was measured in lab.

doi

The DOI for the source, preferred way to identify for source

Type `str` or `None`, default `None`

reference

The long form description of the source if no DOI is available, or more information is needed or wanted.

Type `str`

`__init__(doi="", reference="")`

Constructs a new MeasurementSource object.

Parameters

- **doi** (*str or None, default None*) – The DOI for the source, preferred way to identify for source
- **reference** (*str*) – The long form description of the source if no DOI is available, or more information is needed or wanted.

Methods

<code>__init__([doi, reference])</code>	Constructs a new MeasurementSource object.
<code>json()</code>	Creates a JSON representation of this class.

Continued on next page

Table 28 – continued from previous page

<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.
--	--

json()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

classmethod parse_json (*string_contents*, *encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

CalculationSource

class `propertyestimator.properties.CalculationSource` (*fidelity=None, provenance=None*)

Contains any metadata about how a physical property was calculated.

This includes at which fidelity the property was calculated at (e.g Direct simulation, reweighting, ...) in addition to the parameters which were used as part of the calculations.

fidelity

The fidelity at which the property was calculated

Type `str`

provenance

A dictionary containing information about how the property was calculated.

Type dict of str and Any

__init__ (*fidelity=None, provenance=None*)

Constructs a new CalculationSource object.

Parameters

- **fidelity** (*str*) – The fidelity at which the property was calculated
- **provenance** (*dict of str and Any*) – A dictionary containing information about how the property was calculated.

Methods

<code>__init__([fidelity, provenance])</code>	Constructs a new CalculationSource object.
<code>json()</code>	Creates a JSON representation of this class.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.

json()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

classmethod parse_json (*string_contents*, *encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

Built-in Properties

<i>Density</i>	A class representation of a density property
<i>ExcessMolarVolume</i>	A class representation of an excess molar volume property
<i>DielectricConstant</i>	A class representation of a dielectric property
<i>EnthalpyOfMixing</i>	A class representation of an enthalpy of mixing property
<i>EnthalpyOfVaporization</i>	A class representation of an enthalpy of vaporization property
<i>HostGuestBindingAffinity</i>	A class representation of a host-guest binding affinity property

Density

class `propertyestimator.properties.Density` (*thermodynamic_state=None*, *phase=*, *substance=None*, *value=None*, *uncertainty=None*, *gradients=None*, *source=None*)

A class representation of a density property

__init__ (*thermodynamic_state=None*, *phase=*, *substance=None*, *value=None*, *uncertainty=None*, *gradients=None*, *source=None*)

Constructs a new PhysicalProperty object.

Parameters

- **thermodynamic_state** (`ThermodynamicState`) – The thermodynamic state that the property was measured in.
- **phase** (`PropertyPhase`) – The phase that the property was measured in.
- **substance** (`Substance`) – The composition of the substance that was measured.
- **value** (`unit.Quantity`) – The value of the measured physical property.
- **uncertainty** (`unit.Quantity`) – The uncertainty in the measured value.
- **source** (`Source`) – The source of this property.

Methods

<code>__init__</code> ([thermodynamic_state, phase, ...])	Constructs a new PhysicalProperty object.
<code>get_default_reweighting_workflow_schema</code> (<i>options</i>)	Returns the default workflow to use when estimating this property by reweighting existing data.
<code>get_default_simulation_workflow_schema</code> (<i>options</i>)	Returns the default workflow to use when estimating this property from direct simulations.
<code>get_default_workflow_schema</code> (<i>calculation_layer</i>)	Returns the default workflow schema to use for a specific calculation layer.
<code>json</code> ()	Creates a JSON representation of this class.
<code>parse_json</code> (<i>string_contents</i> [, <i>encoding</i>])	Parses a typed json string into the corresponding class structure.
<code>set_value</code> (<i>value</i> , <i>uncertainty</i>)	Set the value and uncertainty of this property.

Attributes

<code>metadata</code>	Additional metadata associated with this property, such as file paths to coordinate files or ...
<code>multi_component_property</code>	
<code>pressure</code>	The pressure at which the property was collected.
<code>required_data_class</code>	
<code>temperature</code>	The temperature at which the property was collected.

static `get_default_workflow_schema` (*calculation_layer*, *options=None*)

Returns the default workflow schema to use for a specific calculation layer.

Parameters

- **calculation_layer** (*str*) – The calculation layer which will attempt to execute the workflow defined by this schema.
- **options** (*WorkflowOptions*) – The options to use when setting up the default workflows.

Returns The default workflow schema.

Return type *WorkflowSchema*

static `get_default_simulation_workflow_schema` (*options=None*)

Returns the default workflow to use when estimating this property from direct simulations.

Parameters **options** (*WorkflowOptions*) – The default options to use when setting up the estimation workflow.

Returns The schema to follow when estimating this property.

Return type *WorkflowSchema*

static `get_default_reweighting_workflow_schema` (*options*)

Returns the default workflow to use when estimating this property by reweighting existing data.

Parameters **options** (*WorkflowOptions*) – The default options to use when setting up the estimation workflow.

Returns The schema to follow when estimating this property.

Return type *WorkflowSchema*

`json` ()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

property metadata

Additional metadata associated with this property, such as file paths to coordinate files or ...

All property metadata will be made accessible to property estimation workflows.

Type dict of str and Any

classmethod parse_json (*string_contents*, *encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str* or *bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

property pressure

The pressure at which the property was collected.

Type `propertyestimator.unit.Quantity` or `None`

set_value (*value*, *uncertainty*)

Set the value and uncertainty of this property.

Parameters

- **value** (*propertyestimator.unit.Quantity*) – The value of the property.
- **uncertainty** (*propertyestimator.unit.Quantity*) – The uncertainty in the properties value.

property temperature

The temperature at which the property was collected.

Type `propertyestimator.unit.Quantity` or `None`

ExcessMolarVolume

```
class propertyestimator.properties.ExcessMolarVolume (thermodynamic_state=None,
                                                    phase=, substance=None,
                                                    value=None, uncertainty=None,
                                                    gradients=None,
                                                    source=None)
```

A class representation of an excess molar volume property

```
__init__ (thermodynamic_state=None, phase=, substance=None, value=None, uncertainty=None,
          gradients=None, source=None)
```

Constructs a new PhysicalProperty object.

Parameters

- **thermodynamic_state** (`ThermodynamicState`) – The thermodynamic state that the property was measured in.
- **phase** (`PropertyPhase`) – The phase that the property was measured in.
- **substance** (`Substance`) – The composition of the substance that was measured.

- **value** (*unit.Quantity*) – The value of the measured physical property.
- **uncertainty** (*unit.Quantity*) – The uncertainty in the measured value.
- **source** (*Source*) – The source of this property.

Methods

<code>__init__</code> ([thermodynamic_state, phase, ...])	Constructs a new PhysicalProperty object.
<code>get_default_reweighting_workflow_schema</code> (<i>options</i>)	Returns the default workflow to use when estimating this property by reweighting existing data.
<code>get_default_simulation_workflow_schema</code> (<i>options</i>)	Returns the default workflow to use when estimating this property from direct simulations.
<code>get_default_workflow_schema</code> (<i>calculation_layer</i>)	Returns the default workflow schema to use for a specific calculation layer.
<code>json</code> ()	Creates a JSON representation of this class.
<code>parse_json</code> (<i>string_contents</i> [, <i>encoding</i>])	Parses a typed json string into the corresponding class structure.
<code>set_value</code> (<i>value</i> , <i>uncertainty</i>)	Set the value and uncertainty of this property.

Attributes

<code>metadata</code>	Additional metadata associated with this property, such as file paths to coordinate files or ...
<code>multi_component_property</code>	
<code>pressure</code>	The pressure at which the property was collected.
<code>required_data_class</code>	
<code>temperature</code>	The temperature at which the property was collected.

static `get_default_workflow_schema` (*calculation_layer*, *options=None*)

Returns the default workflow schema to use for a specific calculation layer.

Parameters

- **calculation_layer** (*str*) – The calculation layer which will attempt to execute the workflow defined by this schema.
- **options** (*WorkflowOptions*) – The options to use when setting up the default workflows.

Returns The default workflow schema.

Return type *WorkflowSchema*

static `get_default_simulation_workflow_schema` (*options=None*)

Returns the default workflow to use when estimating this property from direct simulations.

Parameters **options** (*WorkflowOptions*) – The default options to use when setting up the estimation workflow.

Returns The schema to follow when estimating this property.

Return type *WorkflowSchema*

static `get_default_reweighting_workflow_schema` (*options=None*)

Returns the default workflow to use when estimating this property by reweighting existing data.

Parameters `options` (`WorkflowOptions`) – The default options to use when setting up the estimation workflow.

Returns The schema to follow when estimating this property.

Return type `WorkflowSchema`

`json()`

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

property metadata

Additional metadata associated with this property, such as file paths to coordinate files or ...

All property metadata will be made accessible to property estimation workflows.

Type dict of str and Any

classmethod `parse_json` (`string_contents`, `encoding='utf8'`)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (`str` or `bytes`) – The typed json string.
- **encoding** (`str`) – The encoding of the `string_contents`.

Returns The parsed class.

Return type Any

property pressure

The pressure at which the property was collected.

Type `propertyestimator.unit.Quantity` or `None`

set_value (`value`, `uncertainty`)

Set the value and uncertainty of this property.

Parameters

- **value** (`propertyestimator.unit.Quantity`) – The value of the property.
- **uncertainty** (`propertyestimator.unit.Quantity`) – The uncertainty in the properties value.

property temperature

The temperature at which the property was collected.

Type `propertyestimator.unit.Quantity` or `None`

DielectricConstant

class `propertyestimator.properties.DielectricConstant` (`thermodynamic_state=None`, `phase=`, `substance=None`, `value=None`, `uncertainty=None`, `gradients=None`, `source=None`)

A class representation of a dielectric property

`__init__` (*thermodynamic_state=None, phase=, substance=None, value=None, uncertainty=None, gradients=None, source=None*)
 Constructs a new PhysicalProperty object.

Parameters

- **thermodynamic_state** (*ThermodynamicState*) – The thermodynamic state that the property was measured in.
- **phase** (*PropertyPhase*) – The phase that the property was measured in.
- **substance** (*Substance*) – The composition of the substance that was measured.
- **value** (*unit.Quantity*) – The value of the measured physical property.
- **uncertainty** (*unit.Quantity*) – The uncertainty in the measured value.
- **source** (*Source*) – The source of this property.

Methods

<code>__init__</code> ([<i>thermodynamic_state, phase, ...</i>])	Constructs a new PhysicalProperty object.
<code>get_default_reweighting_workflow_schema</code> (<i>options=None</i>)	Returns the default workflow to use when estimating this property by reweighting existing data.
<code>get_default_simulation_workflow_schema</code> (<i>options=None</i>)	Returns the default workflow to use when estimating this property from direct simulations.
<code>get_default_workflow_schema</code> (<i>calculation_layer, options=None</i>)	Returns the default workflow schema to use for a specific calculation layer.
<code>json</code> ()	Creates a JSON representation of this class.
<code>parse_json</code> (<i>string_contents[, encoding]</i>)	Parses a typed json string into the corresponding class structure.
<code>set_value</code> (<i>value, uncertainty</i>)	Set the value and uncertainty of this property.

Attributes

<code>metadata</code>	Additional metadata associated with this property, such as file paths to coordinate files or ...
<code>multi_component_property</code>	
<code>pressure</code>	The pressure at which the property was collected.
<code>required_data_class</code>	
<code>temperature</code>	The temperature at which the property was collected.

static `get_default_workflow_schema` (*calculation_layer, options=None*)
 Returns the default workflow schema to use for a specific calculation layer.

Parameters

- **calculation_layer** (*str*) – The calculation layer which will attempt to execute the workflow defined by this schema.
- **options** (*WorkflowOptions*) – The options to use when setting up the default workflows.

Returns The default workflow schema.

Return type *WorkflowSchema*

static `get_default_simulation_workflow_schema` (*options=None*)

Returns the default workflow to use when estimating this property from direct simulations.

Parameters `options` (`WorkflowOptions`) – The default options to use when setting up the estimation workflow.

Returns The schema to follow when estimating this property.

Return type `WorkflowSchema`

static `get_default_reweighting_workflow_schema` (*options=None*)

Returns the default workflow to use when estimating this property by reweighting existing data.

Parameters `options` (`WorkflowOptions`) – The default options to use when setting up the estimation workflow.

Returns The schema to follow when estimating this property.

Return type `WorkflowSchema`

json ()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

property metadata

Additional metadata associated with this property, such as file paths to coordinate files or ...

All property metadata will be made accessible to property estimation workflows.

Type dict of str and Any

classmethod `parse_json` (*string_contents, encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

property pressure

The pressure at which the property was collected.

Type `propertyestimator.unit.Quantity` or `None`

set_value (*value, uncertainty*)

Set the value and uncertainty of this property.

Parameters

- **value** (*propertyestimator.unit.Quantity*) – The value of the property.
- **uncertainty** (*propertyestimator.unit.Quantity*) – The uncertainty in the properties value.

property temperature

The temperature at which the property was collected.

Type `propertyestimator.unit.Quantity` or `None`

EnthalpyOfMixing

class propertyestimator.properties.**EnthalpyOfMixing** (*thermodynamic_state=None, phase=, substance=None, value=None, uncertainty=None, gradients=None, source=None*)

A class representation of an enthalpy of mixing property

__init__ (*thermodynamic_state=None, phase=, substance=None, value=None, uncertainty=None, gradients=None, source=None*)
 Constructs a new PhysicalProperty object.

Parameters

- **thermodynamic_state** (*ThermodynamicState*) – The thermodynamic state that the property was measured in.
- **phase** (*PropertyPhase*) – The phase that the property was measured in.
- **substance** (*Substance*) – The composition of the substance that was measured.
- **value** (*unit.Quantity*) – The value of the measured physical property.
- **uncertainty** (*unit.Quantity*) – The uncertainty in the measured value.
- **source** (*Source*) – The source of this property.

Methods

<code>__init__</code> ([<i>thermodynamic_state, phase, ...</i>])	Constructs a new PhysicalProperty object.
<code>get_default_reweighting_workflow_schema</code> (<i>options</i>)	Returns the default workflow to use when estimating this property by reweighting existing data.
<code>get_default_simulation_workflow_schema</code> (<i>options</i>)	Returns the default workflow to use when estimating this property from direct simulations.
<code>get_default_workflow_schema</code> (<i>calculation_layer</i>)	Returns the default workflow schema to use for a specific calculation layer.
<code>json</code> ()	Creates a JSON representation of this class.
<code>parse_json</code> (<i>string_contents</i> [, <i>encoding</i>])	Parses a typed json string into the corresponding class structure.
<code>set_value</code> (<i>value, uncertainty</i>)	Set the value and uncertainty of this property.

Attributes

<code>metadata</code>	Additional metadata associated with this property, such as file paths to coordinate files or ...
<code>multi_component_property</code>	
<code>pressure</code>	The pressure at which the property was collected.
<code>required_data_class</code>	
<code>temperature</code>	The temperature at which the property was collected.

EnthalpyWorkflow

alias of EnthalpySchema

static `get_default_workflow_schema` (*calculation_layer, options=None*)
 Returns the default workflow schema to use for a specific calculation layer.

Parameters

- **calculation_layer** (*str*) – The calculation layer which will attempt to execute the workflow defined by this schema.
- **options** (*WorkflowOptions*) – The options to use when setting up the default workflows.

Returns The default workflow schema.

Return type *WorkflowSchema*

static get_default_simulation_workflow_schema (*options=None*)

Returns the default workflow to use when estimating this property from direct simulations.

Parameters **options** (*WorkflowOptions*) – The default options to use when setting up the estimation workflow.

Returns The schema to follow when estimating this property.

Return type *WorkflowSchema*

static get_default_reweighting_workflow_schema (*options=None*)

Returns the default workflow to use when estimating this property by reweighting existing data.

Parameters **options** (*WorkflowOptions*) – The default options to use when setting up the estimation workflow.

Returns The schema to follow when estimating this property.

Return type *WorkflowSchema*

json ()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type *str*

property metadata

Additional metadata associated with this property, such as file paths to coordinate files or ...

All property metadata will be made accessible to property estimation workflows.

Type dict of *str* and *Any*

classmethod parse_json (*string_contents, encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type *Any*

property pressure

The pressure at which the property was collected.

Type *propertyestimator.unit.Quantity* or *None*

set_value (*value, uncertainty*)

Set the value and uncertainty of this property.

Parameters

- **value** (*propertyestimator.unit.Quantity*) – The value of the property.
- **uncertainty** (*propertyestimator.unit.Quantity*) – The uncertainty in the properties value.

property temperature

The temperature at which the property was collected.

Type *propertyestimator.unit.Quantity* or *None*

EnthalpyOfVaporization

```
class propertyestimator.properties.EnthalpyOfVaporization (thermodynamic_state=None,
                                                           phase=,           sub-
                                                           stance=None,
                                                           value=None,       un-
                                                           certainty=None,
                                                           gradients=None,
                                                           source=None)
```

A class representation of an enthalpy of vaporization property

```
__init__ (thermodynamic_state=None, phase=, substance=None, value=None, uncertainty=None,
          gradients=None, source=None)
Constructs a new PhysicalProperty object.
```

Parameters

- **thermodynamic_state** (*ThermodynamicState*) – The thermodynamic state that the property was measured in.
- **phase** (*PropertyPhase*) – The phase that the property was measured in.
- **substance** (*Substance*) – The composition of the substance that was measured.
- **value** (*unit.Quantity*) – The value of the measured physical property.
- **uncertainty** (*unit.Quantity*) – The uncertainty in the measured value.
- **source** (*Source*) – The source of this property.

Methods

<code>__init__([thermodynamic_state, phase, ...])</code>	Constructs a new PhysicalProperty object.
<code>get_default_reweighting_workflow_schema(options)</code>	Returns the default workflow to use when estimating this property by reweighting existing data.
<code>get_default_simulation_workflow_schema(options)</code>	Returns the default workflow to use when estimating this property from direct simulations.
<code>get_default_workflow_schema(calculation_layer)</code>	Returns the default workflow schema to use for a specific calculation layer.
<code>json()</code>	Creates a JSON representation of this class.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.
<code>set_value(value, uncertainty)</code>	Set the value and uncertainty of this property.

Attributes

<i>metadata</i>	Additional metadata associated with this property, such as file paths to coordinate files or ...
<i>multi_component_property</i>	Returns whether this property is dependant on properties of the full mixed substance, or whether it is also dependant on the properties of the individual components also.
<i>pressure</i>	The pressure at which the property was collected.
<i>required_data_class</i>	
<i>temperature</i>	The temperature at which the property was collected.

property multi_component_property

Returns whether this property is dependant on properties of the full mixed substance, or whether it is also dependant on the properties of the individual components also.

static get_default_workflow_schema (*calculation_layer*, *options=None*)

Returns the default workflow schema to use for a specific calculation layer.

Parameters

- **calculation_layer** (*str*) – The calculation layer which will attempt to execute the workflow defined by this schema.
- **options** (*WorkflowOptions*) – The options to use when setting up the default workflows.

Returns The default workflow schema.

Return type *WorkflowSchema*

static get_default_simulation_workflow_schema (*options=None*)

Returns the default workflow to use when estimating this property from direct simulations.

Parameters **options** (*WorkflowOptions*) – The default options to use when setting up the estimation workflow.

Returns The schema to follow when estimating this property.

Return type *WorkflowSchema*

static get_default_reweighting_workflow_schema (*options*)

Returns the default workflow to use when estimating this property by reweighting existing data.

Parameters **options** (*WorkflowOptions*) – The default options to use when setting up the estimation workflow.

Returns The schema to follow when estimating this property.

Return type *WorkflowSchema*

json()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type *str*

property metadata

Additional metadata associated with this property, such as file paths to coordinate files or ...

All property metadata will be made accessible to property estimation workflows.

Type dict of str and Any

classmethod `parse_json` (*string_contents*, *encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str* or *bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

property `pressure`

The pressure at which the property was collected.

Type `propertyestimator.unit.Quantity` or `None`

set_value (*value*, *uncertainty*)

Set the value and uncertainty of this property.

Parameters

- **value** (`propertyestimator.unit.Quantity`) – The value of the property.
- **uncertainty** (`propertyestimator.unit.Quantity`) – The uncertainty in the properties value.

property `temperature`

The temperature at which the property was collected.

Type `propertyestimator.unit.Quantity` or `None`

HostGuestBindingAffinity

class `propertyestimator.properties.HostGuestBindingAffinity` (*thermodynamic_state=None*,
phase=, *sub-*
stance=None,
value=None, *un-*
certainty=None,
gradients=None,
source=None)

A class representation of a host-guest binding affinity property

__init__ (*thermodynamic_state=None*, *phase=*, *substance=None*, *value=None*, *uncertainty=None*,
gradients=None, *source=None*)

Constructs a new PhysicalProperty object.

Parameters

- **thermodynamic_state** (`ThermodynamicState`) – The thermodynamic state that the property was measured in.
- **phase** (`PropertyPhase`) – The phase that the property was measured in.
- **substance** (`Substance`) – The composition of the substance that was measured.
- **value** (`unit.Quantity`) – The value of the measured physical property.
- **uncertainty** (`unit.Quantity`) – The uncertainty in the measured value.
- **source** (`Source`) – The source of this property.

Methods

<code>__init__</code> ([thermodynamic_state, phase, ...])	Constructs a new PhysicalProperty object.
<code>get_default_simulation_workflow_schema</code> (<i>options</i>)	Returns the default workflow to use when estimating this property from direct simulations.
<code>get_default_workflow_schema</code> (<i>calculation_layer</i>)	Returns the default workflow schema to use for a specific calculation layer.
<code>json</code> ()	Creates a JSON representation of this class.
<code>parse_json</code> (<i>string_contents</i> [, <i>encoding</i>])	Parses a typed json string into the corresponding class structure.
<code>set_value</code> (<i>value</i> , <i>uncertainty</i>)	Set the value and uncertainty of this property.

Attributes

<code>metadata</code>	Additional metadata associated with this property, such as file paths to coordinate files or ...
<code>multi_component_property</code>	Returns whether this property is dependant on properties of the full mixed substance, or whether it is also dependant on the properties of the individual components also.
<code>pressure</code>	The pressure at which the property was collected.
<code>temperature</code>	The temperature at which the property was collected.

`property multi_component_property`

Returns whether this property is dependant on properties of the full mixed substance, or whether it is also dependant on the properties of the individual components also.

`static get_default_workflow_schema` (*calculation_layer*, *options=None*)

Returns the default workflow schema to use for a specific calculation layer.

Parameters

- **calculation_layer** (*str*) – The calculation layer which will attempt to execute the workflow defined by this schema.
- **options** (*WorkflowOptions*) – The options to use when setting up the default workflows.

Returns The default workflow schema.

Return type *WorkflowSchema*

`static get_default_simulation_workflow_schema` (*options=None*)

Returns the default workflow to use when estimating this property from direct simulations.

Parameters **options** (*WorkflowOptions*) – The default options to use when setting up the estimation workflow.

Returns The schema to follow when estimating this property.

Return type *WorkflowSchema*

`json` ()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type *str*

property metadata

Additional metadata associated with this property, such as file paths to coordinate files or ...

All property metadata will be made accessible to property estimation workflows.

Type dict of str and Any

classmethod parse_json (*string_contents*, *encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str* or *bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

property pressure

The pressure at which the property was collected.

Type propertyestimator.unit.Quantity or None

set_value (*value*, *uncertainty*)

Set the value and uncertainty of this property.

Parameters

- **value** (*propertyestimator.unit.Quantity*) – The value of the property.
- **uncertainty** (*propertyestimator.unit.Quantity*) – The uncertainty in the properties value.

property temperature

The temperature at which the property was collected.

Type propertyestimator.unit.Quantity or None

Substance Definition

<i>Substance</i>	Defines the components, their amounts, and their roles in a system.
------------------	---

Substance

class propertyestimator.substances.**Substance**

Defines the components, their amounts, and their roles in a system.

Examples

A neat liquid containing only a single component:

```
>>> liquid = Substance()
>>> liquid.add_component(Substance.Component(smiles='O'), Substance.
↳MoleFraction(1.0))
```

A binary mixture containing two components, where the mole fractions are explicitly stated:

```
>>> binary_mixture = Substance()
>>> binary_mixture.add_component(Substance.Component(smiles='O'), Substance.
↳MoleFraction(0.2))
>>> binary_mixture.add_component(Substance.Component(smiles='CO'), Substance.
↳MoleFraction(0.8))
```

The infinite dilution of one molecule within a bulk solvent or mixture may also be specified by defining the exact number of copies of that molecule, rather than a mole fraction:

```
>>> benzene = Substance.Component(smiles='C1=CC=CC=C1', role=Substance.
↳ComponentRole.Solute)
>>> water = Substance.Component(smiles='O', role=Substance.ComponentRole.Solvent)
>>>
>>> infinite_dilution = Substance()
>>> infinite_dilution.add_component(component=benzene, amount=Substance.
↳ExactAmount(1)) # Infinite dilution.
>>> infinite_dilution.add_component(component=water, amount=Substance.
↳MoleFraction(1.0))
```

In this example we explicitly flag benzene as being the solute and the water component the solvent. This enables workflow's to easily identify key molecules of interest, such as the molecule which should be 'grown' into solution during solvation free energy calculations.

`__init__()`
Constructs a new Substance object.

Methods

<code>__init__()</code>	Constructs a new Substance object.
<code>add_component(component, amount)</code>	Add a component to the Substance.
<code>calculate_aqueous_ionic_mole_fraction</code>	Determines what mole fraction of ions is needed to yield
<code>from_components(*components)</code>	Creates a new <i>Substance</i> object from a list of components.
<code>get_amounts(component)</code>	Returns the amounts of the component in this substance.
<code>get_molecules_per_component(maximum_molecules)</code>	Returns the number of molecules for each component in this substance, given a maximum total number of molecules.
<code>json()</code>	Creates a JSON representation of this class.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.

Attributes

<code>components</code>	A list of all of the components in this substance.
<code>identifier</code>	A unique str representation of this substance, which encodes all components and their amounts in the substance.
<code>number_of_components</code>	The number of different components in this substance.

class ComponentRole

An enum which describes the role of a component in the system, such as whether the component is a solvent, a solute, a receptor etc.

These roles are mainly only used by specific protocols to identify the correct species in a system, such as when doing docking or performing solvation free energy calculations.

class Component (*smiles=None, label=None, role=None*)

Defines a single component in a system, as well as properties such as it's relative proportion in the system.

property identifier

A unique identifier for this component, which is either a smiles descriptor or the supplied label.

Type *str*

property label

A string label which describes this compound, for example, CB8.

Type *str*

property smiles

The smiles pattern which describes this component, which may be None for complex (e.g protein) molecules.

Type *str*

property role

The role of this component in the system, such as a ligand or a receptor.

Type *ComponentRole*

json()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type *str*

classmethod parse_json (*string_contents, encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type *Any*

class Amount (*value=None*)

An abstract representation of the amount of a given component in a substance.

property value

The value of this amount.

property identifier

A string identifier for this amount.

abstract to_number_of_molecules (*total_substance_molecules, tolerance=None*)

Converts this amount to an exact number of molecules

Parameters

- **total_substance_molecules** (*int*) – The total number of molecules in the whole substance. This amount will contribute to a portion of this total number.
- **tolerance** (*float, optional*) – The tolerance with which this amount should be in. As an example, when converting a mole fraction into a number of molecules, the total number of molecules may not be sufficiently large enough to reproduce this amount.

Returns The number of molecules which this amount represents, given the *total_substance_molecules*.

Return type `int`

class MoleFraction (*value=1.0*)

Represents the amount of a component in a substance as a mole fraction.

property value

The value of this amount.

Type `float`

property identifier

A string identifier for this amount.

to_number_of_molecules (*total_substance_molecules, tolerance=None*)

Converts this amount to an exact number of molecules

Parameters

- **total_substance_molecules** (*int*) – The total number of molecules in the whole substance. This amount will contribute to a portion of this total number.
- **tolerance** (*float, optional*) – The tolerance with which this amount should be in. As an example, when converting a mole fraction into a number of molecules, the total number of molecules may not be sufficiently large enough to reproduce this amount.

Returns The number of molecules which this amount represents, given the *total_substance_molecules*.

Return type `int`

class ExactAmount (*value=1*)

Represents the amount of a component in a substance as an exact number of molecules.

The expectation is that this amount should be used for components which are infinitely dilute (such as ligands in binding calculations), and hence do not contribute to the total mole fraction of a substance

property value

The value of this amount.

Type `int`

property identifier

A string identifier for this amount.

to_number_of_molecules (*total_substance_molecules, tolerance=None*)

Converts this amount to an exact number of molecules

Parameters

- **total_substance_molecules** (*int*) – The total number of molecules in the whole substance. This amount will contribute to a portion of this total number.
- **tolerance** (*float, optional*) – The tolerance with which this amount should be in. As an example, when converting a mole fraction into a number of molecules, the total number of molecules may not be sufficiently large enough to reproduce this amount.

Returns The number of molecules which this amount represents, given the *total_substance_molecules*.

Return type `int`

property identifier

A unique str representation of this substance, which encodes all components and their amounts in the substance.

Type `str`

property components

A list of all of the components in this substance.

Type list of Substance.Component

property number_of_components

The number of different components in this substance.

Type int

classmethod from_components (*components)

Creates a new *Substance* object from a list of components. This method assumes that all components should be present with equal mole fractions.

Parameters **components** (*Substance.Component* or *str*) – The components to add to the substance. These may either be full *Substance.Component* objects or just the smiles representation of the component.

Returns The substance containing the requested components in equal amounts.

Return type *Substance*

add_component (component, amount)

Add a component to the Substance. If the component is already present in the substance, then the mole fraction will be added to the current mole fraction of that component.

Parameters

- **component** (*Substance.Component*) – The component to add to the system.
- **amount** (*Substance.Amount*) – The amount of this component in the substance.

get_amounts (component)

Returns the amounts of the component in this substance.

Parameters **component** (*str* or *Substance.Component*) – The component (or its identifier) to retrieve the amount of.

Returns The amounts of the component in this substance.

Return type list of Substance.Amount

get_molecules_per_component (maximum_molecules, tolerance=None)

Returns the number of molecules for each component in this substance, given a maximum total number of molecules.

Parameters

- **maximum_molecules** (*int*) – The maximum number of molecules.
- **tolerance** (*float*, *optional*) – The tolerance within which this amount should be represented. As an example, when converting a mole fraction into a number of molecules, the total number of molecules may not be sufficiently large enough to reproduce this amount.

Returns A dictionary of molecule counts per component, where each key is a component identifier.

Return type dict of str and int

static calculate_aqueous_ionic_mole_fraction (ionic_strength)

Determines what mole fraction of ions is needed to yield an aqueous system of a given ionic strength.

Parameters **ionic_strength** (*unit.Quantity*) – The ionic string in units of molar.

Returns The mole fraction of ions.

Return type `float`

`json()`

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

classmethod `parse_json` (*string_contents*, *encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

State Definition

ThermodynamicState

Data specifying a physical thermodynamic state obeying Boltzmann statistics.

ThermodynamicState

class `propertyestimator.thermodynamics.ThermodynamicState` (*temperature=None*,
pressure=None)

Data specifying a physical thermodynamic state obeying Boltzmann statistics.

temperature

The external temperature

Type `propertyestimator.unit.Quantity with units compatible with kelvin`

pressure

The external pressure

Type `propertyestimator.unit.Quantity with units compatible with atmospheres`

Examples

Specify an NPT state at 298 K and 1 atm pressure.

```
>>> state = ThermodynamicState(temperature=298.0*unit.kelvin, pressure=1.0*unit.
↪atmospheres)
```

Note that the pressure is only relevant for periodic systems.

__init__ (*temperature=None*, *pressure=None*)

Constructs a new `ThermodynamicState` object.

Parameters

- **temperature** (*propertyestimator.unit.Quantity with units compatible with kelvin*) – The external temperature

- **pressure** (*propertyestimator.unit.Quantity with units compatible with atmospheres*) – The external pressure

Methods

<code>__init__</code> ([temperature, pressure])	Constructs a new ThermodynamicState object.
<code>json</code> ()	Creates a JSON representation of this class.
<code>parse_json</code> (string_contents[, encoding])	Parses a typed json string into the corresponding class structure.

Attributes

<code>beta</code>	Returns one divided by the temperature multiplied by the molar gas constant
<code>inverse_beta</code>	Returns the temperature multiplied by the molar gas constant

property inverse_beta

Returns the temperature multiplied by the molar gas constant

property beta

Returns one divided by the temperature multiplied by the molar gas constant

json ()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

classmethod parse_json (*string_contents, encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

Metadata

<code>PropertyPhase</code>	An enum describing the phase a property was collected in.
<code>Source</code>	Container class for information about how a property was measured / calculated.
<code>MeasurementSource</code>	Contains any metadata about how a physical property was measured by experiment.
<code>CalculationSource</code>	Contains any metadata about how a physical property was calculated.

1.5.4 Data Set API

<i>PhysicalPropertyDataSet</i>	An object for storing and curating data sets of both physical property measurements and estimated.
--------------------------------	--

PhysicalPropertyDataSet

class propertyestimator.datasets.**PhysicalPropertyDataSet**

An object for storing and curating data sets of both physical property measurements and estimated. This class defines a number of convenience functions for filtering out unwanted properties, and for generating general statistics (such as the number of properties per substance) about the set.

`__init__()`

Constructs a new PhysicalPropertyDataSet object.

Methods

<code>__init__()</code>	Constructs a new PhysicalPropertyDataSet object.
<code>filter_by_components(number_of_components)</code>	Filter the data set based on a minimum and maximum temperature.
<code>filter_by_elements(*allowed_elements)</code>	Filters out those properties which were estimated for
<code>filter_by_function(filter_function)</code>	Filter the data set using a given filter function.
<code>filter_by_phases(phases)</code>	Filter the data set based on the phase of the property (e.g liquid).
<code>filter_by_pressure(min_pressure, max_pressure)</code>	Filter the data set based on a minimum and maximum pressure.
<code>filter_by_property_types(*property_type)</code>	Filter the data set based on the type of property (e.g Density).
<code>filter_by_smiles(*allowed_smiles)</code>	Filters out those properties which were estimated for
<code>filter_by_temperature(min_temperature, ...)</code>	Filter the data set based on a minimum and maximum temperature.
<code>filter_by_uncertainties()</code>	Filters out those properties which don't have their uncertainties reported.
<code>json()</code>	Creates a JSON representation of this class.
<code>merge(data_set)</code>	Merge another data set into the current one.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.
<code>to_pandas()</code>	Converts a <i>PhysicalPropertyDataSet</i> to a <i>pandas.DataFrame</i> object with columns of

Attributes

<code>number_of_properties</code>	The number of properties in the data set.
<code>properties</code>	A list of all of the properties within this set, partitioned by substance identifier.
<code>sources</code>	The list of sources from which the properties were gathered

property properties

A list of all of the properties within this set, partitioned by substance identifier.

TODO: Add a link to Substance.identifier when have access to sphinx docs. TODO: Investigate why PhysicalProperty is not cross-linking.

See also:

Substance.identifier

Type dict of str and list of PhysicalProperty

property sources

The list of sources from which the properties were gathered

Type list of Source

property number_of_properties

The number of properties in the data set.

Type int

merge (data_set)

Merge another data set into the current one.

Parameters **data_set** (*PhysicalPropertyDataSet*) – The secondary data set to merge into this one.

filter_by_function (filter_function)

Filter the data set using a given filter function.

Parameters **filter_function** (*lambda*) – The filter function.

filter_by_property_types (*property_type)

Filter the data set based on the type of property (e.g Density).

Parameters **property_type** (*PropertyType or str*) – The type of property which should be retained.

Examples

Filter the dataset to only contain densities and static dielectric constants

```
>>> # Load in the data set of properties which will be used for comparisons
>>> from propertyestimator.datasets import ThermoMLDataSet
>>> data_set = ThermoMLDataSet.from_doi('10.1016/j.jct.2016.10.001')
>>>
>>> # Filter the dataset to only include densities and dielectric constants.
>>> from propertyestimator.properties import Density, DielectricConstant
>>> data_set.filter_by_property_types(Density, DielectricConstant)
```

or

```
>>> data_set.filter_by_property_types('Density', 'DielectricConstant')
```

filter_by_phases (phases)

Filter the data set based on the phase of the property (e.g liquid).

Parameters **phases** (*PropertyPhase*) – The phase of property which should be retained.

Examples

Filter the dataset to only include liquid properties.

```
>>> # Load in the data set of properties which will be used for comparisons
>>> from propertyestimator.datasets import ThermoMLDataSet
>>> data_set = ThermoMLDataSet.from_doi('10.1016/j.jct.2016.10.001')
>>>
>>> from propertyestimator.properties import PropertyPhase
>>> data_set.filter_by_temperature(PropertyPhase.Liquid)
```

filter_by_temperature (*min_temperature*, *max_temperature*)

Filter the data set based on a minimum and maximum temperature.

Parameters

- **min_temperature** (*unit.Quantity*) – The minimum temperature.
- **max_temperature** (*unit.Quantity*) – The maximum temperature.

Examples

Filter the dataset to only include properties measured between 130-260 K.

```
>>> # Load in the data set of properties which will be used for comparisons
>>> from propertyestimator.datasets import ThermoMLDataSet
>>> data_set = ThermoMLDataSet.from_doi('10.1016/j.jct.2016.10.001')
>>>
>>> from propertyestimator import unit
>>> data_set.filter_by_temperature(min_temperature=130*unit.kelvin, max_
↳temperature=260*unit.kelvin)
```

filter_by_pressure (*min_pressure*, *max_pressure*)

Filter the data set based on a minimum and maximum pressure.

Parameters

- **min_pressure** (*unit.Quantity*) – The minimum pressure.
- **max_pressure** (*unit.Quantity*) – The maximum pressure.

Examples

Filter the dataset to only include properties measured between 70-150 kPa.

```
>>> # Load in the data set of properties which will be used for comparisons
>>> from propertyestimator.datasets import ThermoMLDataSet
>>> data_set = ThermoMLDataSet.from_doi('10.1016/j.jct.2016.10.001')
>>>
>>> from propertyestimator import unit
>>> data_set.filter_by_temperature(min_pressure=70*unit.kilopascal, max_
↳temperature=150*unit.kilopascal)
```

filter_by_components (*number_of_components*)

Filter the data set based on a minimum and maximum temperature.

- Parameters **number_of_components** (*int*) – The allowed number of components in the mixture.

Examples

Filter the dataset to only include pure substance properties.

```
>>> # Load in the data set of properties which will be used for comparisons
>>> from propertyestimator.datasets import ThermoMLDataSet
>>> data_set = ThermoMLDataSet.from_doi('10.1016/j.jct.2016.10.001')
>>>
>>> data_set.filter_by_components(number_of_components=1)
```

filter_by_elements (*allowed_elements)

Filters out those properties which were estimated for compounds which contain elements outside of those defined in *allowed_elements*.

Parameters *allowed_elements* (*str*) – The symbols (e.g. C, H, Cl) of the elements to retain.

filter_by_smiles (*allowed_smiles)

Filters out those properties which were estimated for compounds which do not appear in the allowed *smiles* list.

Parameters *allowed_smiles* (*str*) – The smiles identifiers of the compounds to keep after filtering.

filter_by_uncertainties ()

Filters out those properties which don't have their uncertainties reported.

to_pandas ()

Converts a *PhysicalPropertyDataSet* to a *pandas.DataFrame* object with columns of

- 'Temperature'
- 'Pressure'
- 'Phase'
- 'Number Of Components'
- 'Component 1'
- 'Mole Fraction 1'
- ...
- 'Component N'
- 'Mole Fraction N'
- '<Property 1> Value'
- '<Property 1> Uncertainty'
- ...
- '<Property N> Value'
- '<Property N> Uncertainty'
- 'Source'

where 'Component X' is a column containing the smiles representation of component X.

Returns The create data frame.

Return type pandas.DataFrame

json()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type str

classmethod parse_json (*string_contents*, *encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str* or *bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

NIST ThermoML Archive

<i>ThermoMLDataSet</i>	A dataset of physical property measurements created from a ThermoML dataset.
<i>register_thermoml_property</i>	A decorator which registers information on how to parse a given ThermoML property

ThermoMLDataSet

class propertyestimator.datasets.**ThermoMLDataSet**

A dataset of physical property measurements created from a ThermoML dataset.

Examples

For example, we can use the DOI *10.1016/j.jct.2005.03.012* as a key for retrieving the dataset from the ThermoML Archive:

```
>>> dataset = ThermoMLDataSet.from_doi('10.1016/j.jct.2005.03.012')
```

You can also specify multiple ThermoML Archive keys to create a dataset from multiple ThermoML files:

```
>>> thermoml_keys = ['10.1021/acs.jced.5b00365', '10.1021/acs.jced.5b00474']
>>> dataset = ThermoMLDataSet.from_doi(*thermoml_keys)
```

__init__()

Constructs a new ThermoMLDataSet object.

Methods

__init__()	Constructs a new ThermoMLDataSet object.
<i>filter_by_components</i> (<i>number_of_components</i>)	Filter the data set based on a minimum and maximum temperature.

Continued on next page

Table 54 – continued from previous page

<code>filter_by_elements(*allowed_elements)</code>	Filters out those properties which were estimated for
<code>filter_by_function(filter_function)</code>	Filter the data set using a given filter function.
<code>filter_by_phases(phases)</code>	Filter the data set based on the phase of the property (e.g liquid).
<code>filter_by_pressure(min_pressure, max_pressure)</code>	Filter the data set based on a minimum and maximum pressure.
<code>filter_by_property_types(*property_type)</code>	Filter the data set based on the type of property (e.g Density).
<code>filter_by_smiles(*allowed_smiles)</code>	Filters out those properties which were estimated for
<code>filter_by_temperature(min_temperature, ...)</code>	Filter the data set based on a minimum and maximum temperature.
<code>filter_by_uncertainties()</code>	Filters out those properties which don't have their uncertainties reported.
<code>from_doi(*doi_list)</code>	Load a ThermoML data set from a list of DOIs
<code>from_file(*file_list)</code>	Load a ThermoML data set from a list of files
<code>from_url(*url_list)</code>	Load a ThermoML data set from a list of URLs
<code>from_xml(xml, source)</code>	Load a ThermoML data set from an xml object.
<code>json()</code>	Creates a JSON representation of this class.
<code>merge(data_set)</code>	Merge another data set into the current one.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.
<code>to_pandas()</code>	Converts a <i>PhysicalPropertyDataSet</i> to a <i>pandas.DataFrame</i> object with columns of

Attributes

<code>number_of_properties</code>	The number of properties in the data set.
<code>properties</code>	A list of all of the properties within this set, partitioned by substance identifier.
<code>sources</code>	The list of sources from which the properties were gathered

classmethod `from_doi (*doi_list)`

Load a ThermoML data set from a list of DOIs

Parameters `doi_list` (*str*) – The list of DOIs to pull data from

Returns The loaded data set.

Return type *ThermoMLDataSet*

classmethod `from_url (*url_list)`

Load a ThermoML data set from a list of URLs

Parameters `url_list` (*str*) – The list of URLs to pull data from

Returns The loaded data set.

Return type *ThermoMLDataSet*

classmethod `from_file (*file_list)`

Load a ThermoML data set from a list of files

Parameters `file_list` (*str*) – The list of files to pull data from

Returns The loaded data set.

Return type *ThermoMLDataSet*

filter_by_components (*number_of_components*)

Filter the data set based on a minimum and maximum temperature.

Parameters **number_of_components** (*int*) – The allowed number of components in the mixture.

Examples

Filter the dataset to only include pure substance properties.

```
>>> # Load in the data set of properties which will be used for comparisons
>>> from propertyestimator.datasets import ThermoMLDataSet
>>> data_set = ThermoMLDataSet.from_doi('10.1016/j.jct.2016.10.001')
>>>
>>> data_set.filter_by_components(number_of_components=1)
```

filter_by_elements (**allowed_elements*)

Filters out those properties which were estimated for compounds which contain elements outside of those defined in *allowed_elements*.

Parameters **allowed_elements** (*str*) – The symbols (e.g. C, H, Cl) of the elements to retain.

filter_by_function (*filter_function*)

Filter the data set using a given filter function.

Parameters **filter_function** (*lambda*) – The filter function.

filter_by_phases (*phases*)

Filter the data set based on the phase of the property (e.g liquid).

Parameters **phases** (*PropertyPhase*) – The phase of property which should be retained.

Examples

Filter the dataset to only include liquid properties.

```
>>> # Load in the data set of properties which will be used for comparisons
>>> from propertyestimator.datasets import ThermoMLDataSet
>>> data_set = ThermoMLDataSet.from_doi('10.1016/j.jct.2016.10.001')
>>>
>>> from propertyestimator.properties import PropertyPhase
>>> data_set.filter_by_temperature(PropertyPhase.Liquid)
```

filter_by_pressure (*min_pressure, max_pressure*)

Filter the data set based on a minimum and maximum pressure.

Parameters

- **min_pressure** (*unit.Quantity*) – The minimum pressure.
- **max_pressure** (*unit.Quantity*) – The maximum pressure.

Examples

Filter the dataset to only include properties measured between 70-150 kPa.

```
>>> # Load in the data set of properties which will be used for comparisons
>>> from propertyestimator.datasets import ThermoMLDataSet
>>> data_set = ThermoMLDataSet.from_doi('10.1016/j.jct.2016.10.001')
>>>
>>> from propertyestimator import unit
>>> data_set.filter_by_temperature(min_pressure=70*unit.kilopascal, max_
↳ temperature=150*unit.kilopascal)
```

filter_by_property_types (**property_type*)

Filter the data set based on the type of property (e.g Density).

Parameters **property_type** (*PropertyType* or *str*) – The type of property which should be retained.

Examples

Filter the dataset to only contain densities and static dielectric constants

```
>>> # Load in the data set of properties which will be used for comparisons
>>> from propertyestimator.datasets import ThermoMLDataSet
>>> data_set = ThermoMLDataSet.from_doi('10.1016/j.jct.2016.10.001')
>>>
>>> # Filter the dataset to only include densities and dielectric constants.
>>> from propertyestimator.properties import Density, DielectricConstant
>>> data_set.filter_by_property_types(Density, DielectricConstant)
```

or

```
>>> data_set.filter_by_property_types('Density', 'DielectricConstant')
```

filter_by_smiles (**allowed_smiles*)

Filters out those properties which were estimated for compounds which do not appear in the allowed *smiles* list.

Parameters **allowed_smiles** (*str*) – The smiles identifiers of the compounds to keep after filtering.

filter_by_temperature (*min_temperature*, *max_temperature*)

Filter the data set based on a minimum and maximum temperature.

Parameters

- **min_temperature** (*unit.Quantity*) – The minimum temperature.
- **max_temperature** (*unit.Quantity*) – The maximum temperature.

Examples

Filter the dataset to only include properties measured between 130-260 K.


```

>>> # Load in the data set of properties which will be used for comparisons
>>> from propertyestimator.datasets import ThermoMLDataSet
>>> data_set = ThermoMLDataSet.from_doi('10.1016/j.jct.2016.10.001')
>>>
>>> from propertyestimator import unit
>>> data_set.filter_by_temperature(min_temperature=130*unit.kelvin, max_
↳temperature=260*unit.kelvin)

```

filter_by_uncertainties()

Filters out those properties which don't have their uncertainties reported.

classmethod from_xml(xml, source)

Load a ThermoML data set from an xml object.

Parameters

- **xml** (*str*) – The xml string to parse.
- **source** (*Source*) – The source of the xml object.

Returns The loaded ThermoML data set.

Return type *ThermoMLDataSet*

json()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type *str*

merge(data_set)

Merge another data set into the current one.

Parameters **data_set** (*PhysicalPropertyDataSet*) – The secondary data set to merge into this one.

property number_of_properties

The number of properties in the data set.

Type *int*

classmethod parse_json(string_contents, encoding='utf8')

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type *Any*

property properties

A list of all of the properties within this set, partitioned by substance identifier.

TODO: Add a link to `Substance.identifier` when have access to sphinx docs. TODO: Investigate why `PhysicalProperty` is not cross-linking.

See also:

`Substance.identifier`

Type dict of str and list of PhysicalProperty

property sources

The list of sources from which the properties were gathered

Type list of Source

to_pandas ()

Converts a *PhysicalPropertyDataSet* to a *pandas.DataFrame* object with columns of

- 'Temperature'
- 'Pressure'
- 'Phase'
- 'Number Of Components'
- 'Component 1'
- 'Mole Fraction 1'
- ...
- 'Component N'
- 'Mole Fraction N'
- '<Property 1> Value'
- '<Property 1> Uncertainty'
- ...
- '<Property N> Value'
- '<Property N> Uncertainty'
- 'Source'

where 'Component X' is a column containing the smiles representation of component X.

Returns The create data frame.

Return type pandas.DataFrame

propertyestimator.datasets.register_thermoml_property

propertyestimator.datasets.**register_thermoml_property** (*thermoml_string*, *supported_phases*)

A decorator which registers information on how to parse a given ThermoML property

For now this only takes input of a thermoML string, but in future will give greater control over exactly how ThermoML XML gets parsed to an actual property.

Parameters

- **thermoml_string** (*str*) – The ThermoML string identifier (ePropName) for this property.
- **supported_phases** (*PropertyPhase:*) – An enum which encodes all of the phases for which this property supports being estimated in.

1.5.5 Calculation Layers API

<code>PropertyCalculationLayer</code>	An abstract representation of a calculation layer whose goal is to estimate a set of physical properties using a single approach, such as a layer which employs direct simulations to estimate properties, or one which reweights cached simulation data to the same end.
<code>register_calculation_layer</code>	A decorator which registers a class as being a calculation layer which may be used in property calculations.

PropertyCalculationLayer

class `propertyestimator.layers.PropertyCalculationLayer`

An abstract representation of a calculation layer whose goal is to estimate a set of physical properties using a single approach, such as a layer which employs direct simulations to estimate properties, or one which reweights cached simulation data to the same end.

Notes

Calculation layers must inherit from this class, and must override the `schedule_calculation` method.

See also:

TODO Link to a general page outlining what calculation layers are and how they are used.

`__init__()`
Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__</code>	Initialize self.
<code>schedule_calculation</code> (<code>calculation_backend</code> , ...)	Submit the proposed calculation to the backend of choice.

static `schedule_calculation`(`calculation_backend`, `storage_backend`, `layer_directory`,
`data_model`, `callback`, `synchronous=False`)
Submit the proposed calculation to the backend of choice.

Parameters

- **calculation_backend** (`PropertyEstimatorBackend`) – The backend to the submit the calculations to.
- **storage_backend** (`PropertyEstimatorStorage`) – The backend used to store / retrieve data from previous calculations.
- **layer_directory** (`str`) – The local directory in which to store all local, temporary calculation data from this layer.
- **data_model** (`PropertyEstimatorServer.ServerEstimationRequest`) – The data model encoding the proposed calculation.
- **callback** (`function`) – The function to call when the backend returns the results (or an error).

- **synchronous** (*bool*) – If true, this function will block until the calculation has completed. This is mainly intended for debugging purposes.

propertyestimator.layers.register_calculation_layer

propertyestimator.layers.**register_calculation_layer** ()

A decorator which registers a class as being a calculation layer which may be used in property calculations.

See also:

TODO () add documentation for plugin support

Built-in Calculation Layers

<i>ReweightingLayer</i>	A calculation layer which aims to calculate physical properties by reweighting the results of previous calculations.
<i>SimulationLayer</i>	A calculation layer which aims to calculate physical properties directly from molecular simulation.

ReweightingLayer

class propertyestimator.layers.**ReweightingLayer**

A calculation layer which aims to calculate physical properties by reweighting the results of previous calculations.

Warning: This class is still heavily under development and is subject to rapid changes.

`__init__` ()

Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__</code>	Initialize self.
<code>schedule_calculation</code> (calculation_backend, ...)	Submit the proposed calculation to the backend of choice.

static `schedule_calculation` (*calculation_backend*, *storage_backend*, *layer_directory*, *data_model*, *callback*, *synchronous=False*)

Submit the proposed calculation to the backend of choice.

Parameters

- **calculation_backend** (*PropertyEstimatorBackend*) – The backend to the submit the calculations to.
- **storage_backend** (*PropertyEstimatorStorage*) – The backend used to store / retrieve data from previous calculations.
- **layer_directory** (*str*) – The local directory in which to store all local, temporary calculation data from this layer.

- **data_model** (`PropertyEstimatorServer.ServerEstimationRequest`) – The data model encoding the proposed calculation.
- **callback** (*function*) – The function to call when the backend returns the results (or an error).
- **synchronous** (*bool*) – If true, this function will block until the calculation has completed. This is mainly intended for debugging purposes.

SimulationLayer

class `propertyestimator.layers.SimulationLayer`

A calculation layer which aims to calculate physical properties directly from molecular simulation.

Warning: This class is experimental and should not be used in a production environment.

`__init__` ()

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__</code>	Initialize self.
<code>schedule_calculation</code> (<i>calculation_backend</i> , ...)	Submit the proposed calculation to the backend of choice.

static `schedule_calculation` (*calculation_backend*, *storage_backend*, *layer_directory*,
data_model, *callback*, *synchronous=False*)

Submit the proposed calculation to the backend of choice.

Parameters

- **calculation_backend** (`PropertyEstimatorBackend`) – The backend to the submit the calculations to.
- **storage_backend** (`PropertyEstimatorStorage`) – The backend used to store / retrieve data from previous calculations.
- **layer_directory** (*str*) – The local directory in which to store all local, temporary calculation data from this layer.
- **data_model** (`PropertyEstimatorServer.ServerEstimationRequest`) – The data model encoding the proposed calculation.
- **callback** (*function*) – The function to call when the backend returns the results (or an error).
- **synchronous** (*bool*) – If true, this function will block until the calculation has completed. This is mainly intended for debugging purposes.

1.5.6 Calculation Backends API

<i>PropertyEstimatorBackend</i>	An abstract base representation of a property estimator backend.
<i>ComputeResources</i>	An object which stores how many of each type of computational resource (threads or gpu's) is available to a calculation worker.
<i>QueueWorkerResources</i>	An extended resource object with properties specific to calculations which will run on queue based resources, such as LSF, PBS or SLURM.

PropertyEstimatorBackend

class `propertyestimator.backends.PropertyEstimatorBackend` (*number_of_workers=1*,
re-
sources_per_worker=<propertyestimator.backends
object>)

An abstract base representation of a property estimator backend. A backend is responsible for coordinating, distributing and running calculations on the available hardware. This may range from a single machine to a multinode cluster, but *not* across multiple cluster or physical locations.

Notes

All estimator backend classes must inherit from this class, and must implement the *start*, *stop*, and *submit_task* method.

__init__ (*number_of_workers=1*, *resources_per_worker=<propertyestimator.backends.backends.ComputeResources object>*)

Constructs a new PropertyEstimatorBackend object.

Parameters

- **number_of_workers** (*int*) – The number of works to run the calculations on. One worker can perform a single task (e.g run a simulation) at once.
- **resources_per_worker** (*ComputeResources*) – The number of resources to request per worker.

Methods

<i>__init__</i> (<i>[number_of_workers, ...]</i>)	Constructs a new PropertyEstimatorBackend object.
<i>start</i> ()	Start the calculation backend.
<i>stop</i> ()	Stop the calculation backend.
<i>submit_task</i> (<i>function, *args, **kwargs</i>)	Submit a task to the compute resources managed by this backend.

start ()
 Start the calculation backend.

stop ()
 Stop the calculation backend.

submit_task (*function, *args, **kwargs*)
 Submit a task to the compute resources managed by this backend.

Parameters **function** (*function*) – The function to run.

Returns Returns a future object which will eventually point to the results of the submitted task.

Return type Future

ComputeResources

class `propertyestimator.backends.ComputeResources` (*number_of_threads=1*,
number_of_gpus=0, *preferred_gpu_toolkit=None*)

An object which stores how many of each type of computational resource (threads or gpu's) is available to a calculation worker.

TODO: The use of the terminology here is questionable, and is used interchangeable with `process` which may lead to some confusion.

`__init__` (*number_of_threads=1*, *number_of_gpus=0*, *preferred_gpu_toolkit=None*)
Constructs a new `ComputeResources` object.

Parameters

- **`number_of_threads`** (*int*) – The number of threads available to a calculation worker.
- **`number_of_gpus`** (*int*) – The number of GPUs available to a calculation worker.
- **`preferred_gpu_toolkit`** (`ComputeResources.GPUToolkit`, *optional*)
– The preferred toolkit to use when running on GPUs.

Methods

<code>__init__</code> ([<i>number_of_threads</i> , ...])	Constructs a new <code>ComputeResources</code> object.
---	--

Attributes

<code>gpu_device_indices</code>	The indices of the GPUs to run on.
<code>number_of_gpus</code>	The number of GPUs available to a calculation worker.
<code>number_of_threads</code>	The number of threads available to a calculation worker.
<code>preferred_gpu_toolkit</code>	The preferred toolkit to use when running on GPUs.

class `GPUToolkit`

An enumeration of the different GPU toolkits to make available to different calculations.

property `number_of_threads`

The number of threads available to a calculation worker.

Type `int`

property `number_of_gpus`

The number of GPUs available to a calculation worker.

Type `int`

property `preferred_gpu_toolkit`

The preferred toolkit to use when running on GPUs.

Type `ComputeResources.GPUToolkit`

property gpu_device_indices

The indices of the GPUs to run on. This is purely an internal implementation detail and should not be relied upon externally.

Type `str`

QueueWorkerResources

```
class propertyestimator.backends.QueueWorkerResources (number_of_threads=1,
                                                    number_of_gpus=0, preferred_gpu_toolkit=None,
                                                    per_thread_memory_limit=<Quantity(1,
                                                    'gigabyte')>, wallclock_time_limit='01:00')
```

An extended resource object with properties specific to calculations which will run on queue based resources, such as LSF, PBS or SLURM.

```
__init__ (number_of_threads=1, number_of_gpus=0, preferred_gpu_toolkit=None,
          per_thread_memory_limit=<Quantity(1, 'gigabyte')>, wallclock_time_limit='01:00')
```

Constructs a new ComputeResources object.

Notes

Both the requested *number_of_threads* and the *number_of_gpus* must be less than or equal to the number of threads (/cpus/cores) and GPUs available to each compute node in the cluster respectively, such that a single worker is able to be accommodated by a single compute node.

Parameters

- **per_thread_memory_limit** (*simtk.Quantity*) – The maximum amount of memory available to each thread.
- **wallclock_time_limit** (*str*) – The maximum amount of wall clock time that a worker can run for. This should be a string of the form *HH:MM* where HH is the number of hours and MM the number of minutes

Methods

<code>__init__</code> ([number_of_threads, ...])	Constructs a new ComputeResources object.
--	---

Attributes

<code>gpu_device_indices</code>	The indices of the GPUs to run on.
<code>number_of_gpus</code>	The number of GPUs available to a calculation worker.
<code>number_of_threads</code>	The number of threads available to a calculation worker.
<code>per_thread_memory_limit</code>	The maximum amount of memory available to each thread, such that the total memory limit will be <i>per_cpu_memory_limit * number_of_threads</i> .
<code>preferred_gpu_toolkit</code>	The preferred toolkit to use when running on GPUs.

Continued on next page

Table 66 – continued from previous page

<code>wallclock_time_limit</code>	The maximum amount of wall clock time that a worker can run for.
property per_thread_memory_limit	The maximum amount of memory available to each thread, such that the total memory limit will be <code>per_cpu_memory_limit * number_of_threads</code> .
Type	<code>simtk.Quantity</code>
property wallclock_time_limit	The maximum amount of wall clock time that a worker can run for. This should be a string of the form <code>HH:MM</code> where HH is the number of hours and MM the number of minutes
Type	<code>str</code>
class GPUToolkit	An enumeration of the different GPU toolkits to make available to different calculations.
property gpu_device_indices	The indices of the GPUs to run on. This is purely an internal implementation detail and should not be relied upon externally.
Type	<code>str</code>
property number_of_gpus	The number of GPUs available to a calculation worker.
Type	<code>int</code>
property number_of_threads	The number of threads available to a calculation worker.
Type	<code>int</code>
property preferred_gpu_toolkit	The preferred toolkit to use when running on GPUs.
Type	<code>ComputeResources.GPUToolkit</code>

Dask Backends

<code>BaseDaskBackend</code>	A base <i>dask</i> backend class, which implements functionality which is common to all other <i>dask</i> based backends.
<code>DaskLocalCluster</code>	A property estimator backend which uses a <i>dask LocalCluster</i> object to run calculations on a single machine.
<code>DaskLSFBackend</code>	A property estimator backend which uses a <i>dask_jobqueue.LSFCluster</i> object to run calculations within an existing LSF queue.

BaseDaskBackend

class `propertyestimator.backends.BaseDaskBackend` (`number_of_workers=1`, `resources_per_worker=<propertyestimator.backends.backends.object>`)

A base *dask* backend class, which implements functionality which is common to all other *dask* based backends.

`__init__` (*number_of_workers=1, resources_per_worker=<propertyestimator.backends.backends.ComputeResources object>*)
 Constructs a new BaseDaskBackend object.

Methods

<code>__init__</code> ([<i>number_of_workers, ...</i>])	Constructs a new BaseDaskBackend object.
<code>start</code> ()	Start the calculation backend.
<code>stop</code> ()	Stop the calculation backend.
<code>submit_task</code> (<i>function, *args, **kwargs</i>)	Submit a task to the compute resources managed by this backend.

start ()
 Start the calculation backend.

stop ()
 Stop the calculation backend.

submit_task (*function, *args, **kwargs*)
 Submit a task to the compute resources managed by this backend.

Parameters *function* (*function*) – The function to run.

Returns Returns a future object which will eventually point to the results of the submitted task.

Return type Future

DaskLocalCluster

class `propertyestimator.backends.DaskLocalCluster` (*number_of_workers=1, resources_per_worker=<propertyestimator.backends.backends.ComputeResources object>*)

A property estimator backend which uses a *dask LocalCluster* object to run calculations on a single machine.

See also:

`dask.LocalCluster`

`__init__` (*number_of_workers=1, resources_per_worker=<propertyestimator.backends.backends.ComputeResources object>*)
 Constructs a new DaskLocalCluster

Methods

<code>__init__</code> ([<i>number_of_workers, ...</i>])	Constructs a new DaskLocalCluster
<code>start</code> ()	Start the calculation backend.
<code>stop</code> ()	Stop the calculation backend.
<code>submit_task</code> (<i>function, *args, **kwargs</i>)	Submit a task to the compute resources managed by this backend.

start ()
 Start the calculation backend.

submit_task (*function, *args, **kwargs*)

Submit a task to the compute resources managed by this backend.

Parameters `function` (*function*) – The function to run.

Returns Returns a future object which will eventually point to the results of the submitted task.

Return type Future

`stop()`

Stop the calculation backend.

DaskLSFBackend

```
class propertyestimator.backends.DaskLSFBackend (minimum_number_of_workers=1,
                                                maximum_number_of_workers=1, re-
                                                sources_per_worker=<propertyestimator.backends.backends.Q
                                                object>, queue_name='default',
                                                setup_script_commands=None,
                                                extra_script_options=None, adap-
                                                tive_interval='10000ms', dis-
                                                able_nanny_process=False, adap-
                                                tive_class=None)
```

A property estimator backend which uses a `dask_jobqueue.LSFCluster` object to run calculations within an existing LSF queue.

See also:

`dask_jobqueue.LSFCluster`, `DaskPBSBackend`

```
__init__ (minimum_number_of_workers=1, maximum_number_of_workers=1, re-
          sources_per_worker=<propertyestimator.backends.backends.QueueWorkerResources ob-
          ject>, queue_name='default', setup_script_commands=None, extra_script_options=None,
          adaptive_interval='10000ms', disable_nanny_process=False, adaptive_class=None)
```

Constructs a new DaskLSFBackend object

Examples

To create an LSF queueing compute backend which will attempt to spin up workers which have access to a single GPU.

```
>>> # Create a resource object which will request a worker with
>>> # one gpu which will stay alive for five hours.
>>> from propertyestimator.backends import QueueWorkerResources
>>>
>>> resources = QueueWorkerResources(number_of_threads=1,
>>>                                 number_of_gpus=1,
>>>                                 preferred_gpu_
↪ toolkit=QueueWorkerResources.GPUSocketToolkit.CUDA,
>>>                                 wallclock_time_limit='05:00')
>>>
>>> # Define the set of commands which will set up the correct environment
>>> # for each of the workers.
>>> setup_script_commands = [
>>>     'module load cuda/9.2',
>>> ]
>>>
>>> # Define extra options to only run on certain node groups
```

(continues on next page)

(continued from previous page)

```

>>> extra_script_options = [
>>>     '-m "ls-gpu lt-gpu"'
>>> ]
>>>
>>>
>>> # Create the backend which will adaptively try to spin up between one and
>>> # ten workers with the requested resources depending on the calculation.
↪load.
>>> from propertyestimator.backends import DaskLSFBackend
>>>
>>> lsf_backend = DaskLSFBackend(minimum_number_of_workers=1,
>>>                             maximum_number_of_workers=10,
>>>                             resources_per_worker=resources,
>>>                             queue_name='gpuqueue',
>>>                             setup_script_commands=setup_script_commands,
>>>                             extra_script_options=extra_script_options)

```

Methods

<code>__init__</code> ([minimum_number_of_workers, ...])	Constructs a new DaskLSFBackend object
<code>job_script</code> ()	Returns the job script that dask will use to submit workers.
<code>start</code> ()	Start the calculation backend.
<code>stop</code> ()	Stop the calculation backend.
<code>submit_task</code> (function, *args, **kwargs)	Submit a task to the compute resources managed by this backend.

`job_script` ()

Returns the job script that dask will use to submit workers. The backend must be started before calling this function.

Returns

Return type `str`

`start` ()

Start the calculation backend.

`stop` ()

Stop the calculation backend.

`submit_task` (function, *args, **kwargs)

Submit a task to the compute resources managed by this backend.

Parameters `function` (*function*) – The function to run.

Returns Returns a future object which will eventually point to the results of the submitted task.

Return type `Future`

1.5.7 Storage Backends API

*PropertyEstimatorStorage*An abstract base representation of how the property estimator will interact with and store simulation data.

PropertyEstimatorStorage

class `propertyestimator.storage.PropertyEstimatorStorage`

An abstract base representation of how the property estimator will interact with and store simulation data.

Notes

Any inheriting class must provide an implementation for the *store_object*, *retrieve_object* and *has_object* methods

`__init__()`

Constructs a new PropertyEstimatorStorage object.

Methods

<code>__init__()</code>	Constructs a new PropertyEstimatorStorage object.
<code>has_force_field(force_field)</code>	Checks whether the force field has been previously stored in the force field directory.
<code>retrieve_force_field(unique_id)</code>	Retrieves a force field from storage, if it exists.
<code>retrieve_simulation_data(substance[, ...])</code>	Retrieves any data that has been stored for a given substance.
<code>retrieve_simulation_data_by_id(unique_id)</code>	Attempts to retrieve a storage piece of simulation data from it's unique id.
<code>store_force_field(force_field)</code>	Store the force field in the cached force field directory.
<code>store_simulation_data(data_object, ...)</code>	Store the simulation data.

`has_force_field(force_field)`

Checks whether the force field has been previously stored in the force field directory.

Parameters `force_field` (`ForceFieldSource`) – The force field to check for.**Returns** None if the force field has not been cached, otherwise the unique id of the cached force field.**Return type** `str`, optional**`retrieve_force_field(unique_id)`**

Retrieves a force field from storage, if it exists.

Parameters `unique_id` (`str`) – The unique id of the force field to retrieve**Returns** The force field if present in the storage system with the given key, otherwise None.**Return type** `ForceFieldSource`, optional**`store_force_field(force_field)`**

Store the force field in the cached force field directory.

Parameters `force_field` (`ForceFieldSource`) – The force field to store.**Returns** The unique id of the stored force field.

Return type `str`

retrieve_simulation_data_by_id (*unique_id*)

Attempts to retrieve a storage piece of simulation data from its unique id.

Parameters `unique_id` (*str*) – The unique id assigned to the data.

Returns

- `BaseStoredData` – The stored data object.
- *str* – The path to the data's corresponding directory.

retrieve_simulation_data (*substance*, *include_component_data=True*, *data_class=<class 'propertyestimator.storage.dataclasses.StoredSimulationData'>*)

Retrieves any data that has been stored for a given substance.

Parameters

- **substance** (`Substance`) – The substance to check for.
- **include_component_data** (*bool*) – If the substance is a mixture where has multiple components and *include_component_data* is `True`, data will be returned for both the mixed system, and for the individual components, otherwise only data for the mixed system will be returned.
- **data_class** (*subclass of BaseStoredData*) – The type of data to retrieve.

Returns A dictionary of the stored data objects and their corresponding directory paths partitioned by substance id.

Return type dict of `str` and tuple of `BaseStoredData` and `str`

store_simulation_data (*data_object*, *data_directory*)

Store the simulation data.

Notes

If the storage system already contains equivalent information (i.e data stored for the same substance, thermodynamic state and parameter set) then the data will be merged according to the data objects *merge* method.

Parameters

- **data_object** (`BaseStoredData`) – The data object being stored.
- **data_directory** (*str*) – The directory which stores files associated with the data object such as trajectory files.

Returns The unique id of the stored data.

Return type `str`

Built-in Storage Backends

`LocalFileStorage`

A storage backend which stores files in directories on the local disk.

LocalFileStorage

class `propertyestimator.storage.LocalFileStorage` (*root_directory='stored_data'*)
 A storage backend which stores files in directories on the local disk.

`__init__` (*root_directory='stored_data'*)
 Constructs a new PropertyEstimatorStorage object.

Methods

<code>__init__</code> (<i>root_directory</i>)	Constructs a new PropertyEstimatorStorage object.
<code>has_force_field</code> (<i>force_field</i>)	Checks whether the force field has been previously stored in the force field directory.
<code>retrieve_force_field</code> (<i>unique_id</i>)	Retrieves a force field from storage, if it exists.
<code>retrieve_simulation_data</code> (<i>substance</i> , ...)	Retrieves any data that has been stored for a given substance.
<code>retrieve_simulation_data_by_id</code> (<i>unique_id</i>)	Attempts to retrieve a storage piece of simulation data from it's unique id.
<code>store_force_field</code> (<i>force_field</i>)	Store the force field in the cached force field directory.
<code>store_simulation_data</code> (<i>data_object</i> , ...)	Store the simulation data.

Attributes

<code>root_directory</code>	Returns the directory in which all stored objects are located.
-----------------------------	--

property `root_directory`
 Returns the directory in which all stored objects are located.

Type `str`

store_simulation_data (*data_object*, *data_directory*)
 Store the simulation data.

Notes

If the storage system already contains equivalent information (i.e data stored for the same substance, thermodynamic state and parameter set) then the data will be merged according to the data objects *merge* method.

Parameters

- **data_object** (`BaseStoredData`) – The data object being stored.
- **data_directory** (`str`) – The directory which stores files associated with the data object such as trajectory files.

Returns The unique id of the stored data.

Return type `str`

retrieve_simulation_data_by_id (*unique_id*)
 Attempts to retrieve a storage piece of simulation data from it's unique id.

Parameters `unique_id (str)` – The unique id assigned to the data.

Returns

- *BaseStoredData* – The stored data object.
- *str* – The path to the data’s corresponding directory.

retrieve_simulation_data (*substance*, *include_component_data=True*, *data_class=<class 'propertyestimator.storage.dataclasses.StoredSimulationData'>*)

Retrieves any data that has been stored for a given substance.

Parameters

- **substance** (*Substance*) – The substance to check for.
- **include_component_data** (*bool*) – If the substance is a mixture where has multiple components and *include_component_data* is True, data will be returned for both the mixed system, and for the individual components, otherwise only data for the mixed system will be returned.
- **data_class** (*subclass of BaseStoredData*) – The type of data to retrieve.

Returns A dictionary of the stored data objects and their corresponding directory paths partitioned by substance id.

Return type dict of str and tuple of *BaseStoredData* and str

has_force_field (*force_field*)

Checks whether the force field has been previously stored in the force field directory.

Parameters **force_field** (*ForceFieldSource*) – The force field to check for.

Returns None if the force field has not been cached, otherwise the unique id of the cached force field.

Return type str, optional

retrieve_force_field (*unique_id*)

Retrieves a force field from storage, if it exists.

Parameters **unique_id** (*str*) – The unique id of the force field to retrieve

Returns The force field if present in the storage system with the given key, otherwise None.

Return type *ForceFieldSource*, optional

store_force_field (*force_field*)

Store the force field in the cached force field directory.

Parameters **force_field** (*ForceFieldSource*) – The force field to store.

Returns The unique id of the stored force field.

Return type str

Data Classes

<i>BaseStoredData</i>	A base representation of cached data to be stored by a storage backend.
<i>StoredSimulationData</i>	A representation of data which has been cached from a single previous simulation.

Continued on next page

Table 76 – continued from previous page

<i>StoredDataCollection</i>	A collection of stored <i>StoredSimulationData</i> objects, all generated at the same state and using the same force field parameters.
-----------------------------	--

BaseStoredData

class `propertyestimator.storage.dataclasses.BaseStoredData`

A base representation of cached data to be stored by a storage backend.

The expectation is that stored data will exist in storage as two parts:

- 1) A JSON serialized representation of this class (or a subclass), which contains lightweight information such as the state and composition of the system. Any larger pieces of data, such as coordinates or trajectories, should be referenced by this class as a filename.
- 2) A directory like structure (either directly a directory, or some NetCDF like compressed archive) of ancillary files which do not easily lend themselves to be serialized within a JSON object, whose files are referenced by name by the data object.

substance

A description of the composition of the stored system.

Type *Substance*

thermodynamic_state

The state at which the data was collected.

Type *ThermodynamicState*

source_calculation_id

The server id of the calculation which yielded this data.

Type `str`

provenance

A dictionary containing the provenance information about how this data was generated.

Type `dict` of `str` and `Any`

force_field_id

The server assigned unique id of the force field parameters used to generate the data.

Type `str`

`__init__()`

Constructs a new `BaseStoredData` object

Methods

<code>__init__()</code>	Constructs a new <code>BaseStoredData</code> object
<code>can_merge(other_data)</code>	Checks whether this piece of data stores the same amount of compatible information (or more) than another piece of stored data, and hence whether the two can be merged together.
<code>merge(stored_data_1, stored_data_2)</code>	Collapse two pieces of compatible stored data into one.

can_merge (*other_data*)

Checks whether this piece of data stores the same amount of compatible information (or more) than another piece of stored data, and hence whether the two can be merged together.

Parameters **other_data** (*BaseStoredData*) – The other stored data to compare against.

Returns Returns *True* if this piece of data stores the same amount of information or more than another piece of data, or *false* if it contains less or incompatible data.

Return type *bool*

classmethod merge (*stored_data_1*, *stored_data_2*)

Collapse two pieces of compatible stored data into one.

Parameters

- **stored_data_1** (*BaseStoredData*) – The first piece of stored data.
- **stored_data_2** (*BaseStoredData*) – The second piece of stored data.

Returns The merged stored data.

Return type *BaseStoredData*

StoredSimulationData

class `propertyestimator.storage.dataclasses.StoredSimulationData`

A representation of data which has been cached from a single previous simulation.

Notes

The ancillary directory which stores larger information such as trajectories should be of the form:

```
|--- data_object.json
|--- data_directory
|   |--- coordinate_file_name.pdb
|   |--- trajectory_file_name.dcd
|   |--- statistics_file_name.csv
```

coordinate_file_name

The name of a coordinate file which encodes the topology information of the system.

Type *str*

trajectory_file_name

The name of a .dcd trajectory file containing configurations generated by the simulation.

Type *str*

statistics_file_name

The name of a *StatisticsArray* csv file, containing statistics generated by the simulation.

Type *str*

statistical_inefficiency

The statistical inefficiency of the collected data.

Type *float*

total_number_of_molecules

The total number of molecules in the system.

Type int

`__init__()`

Constructs a new `StoredSimulationData` object

Methods

<code>__init__()</code>	Constructs a new <code>StoredSimulationData</code> object
<code>can_merge(other_data)</code>	Checks whether this piece of data stores the same amount of compatible information (or more) than another piece of stored data, and hence whether the two can be merged together.
<code>merge(stored_data_1, stored_data_2)</code>	Collapse two pieces of compatible stored data into one, by only retaining the data with the longest auto-correlation time.

classmethod `merge(stored_data_1, stored_data_2)`

Collapse two pieces of compatible stored data into one, by only retaining the data with the longest auto-correlation time.

Parameters

- **stored_data_1** (`StoredSimulationData`) – The first piece of stored data.
- **stored_data_2** (`StoredSimulationData`) – The second piece of stored data.

Returns The merged stored data.

Return type `StoredSimulationData`

can_merge (`other_data`)

Checks whether this piece of data stores the same amount of compatible information (or more) than another piece of stored data, and hence whether the two can be merged together.

Parameters **other_data** (`BaseStoredData`) – The other stored data to compare against.

Returns Returns `True` if this piece of data stores the same amount of information or more than another piece of data, or `false` if it contains less or incompatible data.

Return type `bool`

StoredDataCollection

class `propertyestimator.storage.dataclasses.StoredDataCollection`

A collection of stored `StoredSimulationData` objects, all generated at the same state and using the same force field parameters.

The ancillary directory which stores larger information such as trajectories should be of the form:

```

|--- data_object.json
|--- data_directory
|   |--- data_key_1
|       |--- coordinate_file_name.pdb
|       |--- trajectory_file_name.dcd
|       |--- statistics_file_name.csv
|   |--- data_key_2
|       |--- coordinate_file_name.pdb

```

(continues on next page)

(continued from previous page)

```

|--- trajectory_file_name.dcd
|--- statistics_file_name.csv
|--- data_key_3
|--- coordinate_file_name.pdb
|--- trajectory_file_name.dcd
|--- statistics_file_name.csv
    
```

data

A dictionary of stored simulation data objects which have been given a unique key.

Type dict of str and StoredSimulationData

__init__()

Constructs a new StoredDataCollection object

Methods

<code>__init__()</code>	Constructs a new StoredDataCollection object
<code>can_merge(other_data_collection)</code>	param other_data_collection The other stored data to compare against.
<code>merge(stored_data_1, stored_data_2)</code>	Collapse two pieces of compatible stored data into one, by only retaining the data with the longest auto-correlation time.

can_merge (*other_data_collection*)

Parameters **other_data_collection** (`StoredDataCollection`) – The other stored data to compare against.

classmethod merge (*stored_data_1, stored_data_2*)

Collapse two pieces of compatible stored data into one, by only retaining the data with the longest auto-correlation time.

Parameters

- **stored_data_1** (`StoredDataCollection`) – The first piece of stored data.
- **stored_data_2** (`StoredDataCollection`) – The second piece of stored data.

Returns The merged stored data.

Return type *StoredDataCollection*

1.5.8 Workflow API

<i>Workflow</i>	Encapsulates and prepares a workflow which is able to estimate a physical property.
<i>WorkflowGraph</i>	A hierarchical structure for storing and submitting the workflows which will estimate a set of physical properties..

Continued on next page

Table 80 – continued from previous page

<i>WorkflowOptions</i>	A set of convenience options used when creating estimation workflows.
<i>IWorkflowProperty</i>	Defines the interface a property must implement to be estimable by a workflow.

Workflow

class `propertyestimator.workflow.Workflow` (*physical_property*, *global_metadata*, *workflow_uuid=None*)

Encapsulates and prepares a workflow which is able to estimate a physical property.

__init__ (*physical_property*, *global_metadata*, *workflow_uuid=None*)

Constructs a new Workflow object.

Parameters

- **physical_property** (*PhysicalProperty*) – The property which this workflow aims to calculate.
- **global_metadata** (*dict of str and Any*) – A dictionary of the global metadata available to each of the workflow properties.
- **workflow_uuid** (*str, optional*) – An optional uuid to assign to this workflow. If none is provided, one will be chosen at random.

Methods

__init__ (<i>physical_property</i> , <i>global_metadata</i>)	Constructs a new Workflow object.
generate_default_metadata (<i>physical_property</i> , ...)	Generates a default global metadata dictionary.
replace_protocol (<i>old_protocol</i> , <i>new_protocol</i>)	Replaces an existing protocol with a new one, while updating all input and local references to point to the new protocol.

Attributes

<code>schema</code>

replace_protocol (*old_protocol*, *new_protocol*)

Replaces an existing protocol with a new one, while updating all input and local references to point to the new protocol.

The main use of this method is when merging multiple protocols into one.

Parameters

- **old_protocol** (*protocols.BaseProtocol or str*) – The protocol (or its id) to replace.
- **new_protocol** (*protocols.BaseProtocol or str*) – The new protocol (or its id) to use.

static generate_default_metadata (*physical_property*, *force_field_path*, *parameter_gradient_keys=None*, *workflow_options=None*)

Generates a default global metadata dictionary.

Parameters

- **physical_property** (*PhysicalProperty*) – The physical property whose arguments are available in the global scope.
- **force_field_path** (*str*) – The path to the force field parameters to use in the workflow.
- **parameter_gradient_keys** (*list of ParameterGradientKey*) – A list of references to all of the parameters which all observables should be differentiated with respect to.
- **workflow_options** (*WorkflowOptions, optional*) – The options provided when an estimate request was submitted.

Returns

The metadata dictionary, with the following keys / types:

- **thermodynamic_state:** *ThermodynamicState* - The state (T,p) at which the property is being computed
- **substance:** *Substance* - The composition of the system of interest.
- **components:** list of *Substance* - The components present in the system for which the property is being estimated.
- **target_uncertainty:** *propertyestimator.unit.Quantity* - The target uncertainty with which properties should be estimated.
- **per_component_uncertainty:** *propertyestimator.unit.Quantity* - The target uncertainty divided by the sqrt of the number of components in the system + 1
- **force_field_path:** *str* - A path to the force field parameters with which the property should be evaluated with.
- **parameter_gradient_keys:** list of *ParameterGradientKey* - A list of references to all of the parameters which all observables should be differentiated with respect to.

Return type dict of str, Any

WorkflowGraph

class `propertyestimator.workflow.WorkflowGraph` (*root_directory=""*)

A hierarchical structure for storing and submitting the workflows which will estimate a set of physical properties..

`__init__` (*root_directory=""*)

Constructs a new WorkflowGraph

Parameters `root_directory` (*str*) – The root directory in which to store all outputs from this graph.

Methods

<code>__init__</code> ([root_directory])	Constructs a new WorkflowGraph
<code>add_workflow</code> (workflow)	Insert a workflow into the workflow graph.
<code>submit</code> (backend[, include_uncertainty_check])	Submits the protocol graph to the backend of choice.

add_workflow (*workflow*)

Insert a workflow into the workflow graph.

Parameters **workflow** (*Workflow*) – The workflow to insert.

submit (*backend, include_uncertainty_check=True*)

Submits the protocol graph to the backend of choice.

Parameters

- **backend** (*PropertyEstimatorBackend*) – The backend to execute the graph on.
- **include_uncertainty_check** (*bool*) – If true, the uncertainty of each estimated property will be checked to ensure it is below the target threshold set in the workflow metadata. If an uncertainty is not included in the workflow metadata, then this parameter will be ignored.

Returns The futures of the submitted protocols.

Return type list of Future

WorkflowOptions

```
class propertyestimator.workflow.WorkflowOptions (convergence_mode=<ConvergenceMode.RelativeUncertainty:
RelativeUncertainty>, relative_uncertainty_fraction=1.0, absolute_uncertainty=None, protocol_replacements=None)
```

A set of convenience options used when creating estimation workflows.

```
__init__ (convergence_mode=<ConvergenceMode.RelativeUncertainty:
RelativeUncertainty>, relative_uncertainty_fraction=1.0, absolute_uncertainty=None, protocol_replacements=None)
```

Constructs a new WorkflowOptions object.

Parameters

- **convergence_mode** (*WorkflowOptions.ConvergenceMode*) – The mode which governs how workflows should decide when they have reached convergence.
- **relative_uncertainty_fraction** (*float, optional*) – If the convergence mode is set to *RelativeUncertainty*, then workflows will by default run simulations until the estimated uncertainty is less than $relative_uncertainty_fraction * property_to_estimate.uncertainty$
- **absolute_uncertainty** (*propertyestimator.unit.Quantity, optional*) – If the convergence mode is set to *AbsoluteUncertainty*, then workflows will by default run simulations until the estimated uncertainty is less than the *absolute_uncertainty*
- **protocol_replacements** (*dict of str and str, optional*) – A dictionary with keys of the types of protocols which should be replaced with those protocols named by the values.

Methods

```
__init__ ([convergence_mode, ...])
```

```
Constructs a new WorkflowOptions object.
```

class ConvergenceMode

The available options for deciding when a workflow has converged. For now, these options include running until the computed uncertainty of a property is within a relative fraction of the measured uncertainty (*ConvergenceMode.RelativeUncertainty*) or is less than some absolute value (*ConvergenceMode.AbsoluteUncertainty*).

IWorkflowProperty

class propertyestimator.workflow.**IWorkflowProperty**

Defines the interface a property must implement to be estimable by a workflow.

__init__()

Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__</code>	Initialize self.
<code>get_default_workflow_schema(...)</code>	

Schema

<i>WorkflowSchema</i>	Outlines the workflow which should be followed when calculating a certain property.
<i>ProtocolSchema</i>	A json serializable representation of a workflow protocol.
<i>ProtocolGroupSchema</i>	A json serializable representation of a workflow protocol group.
<i>ProtocolReplicator</i>	A protocol replicator contains the information necessary to replicate parts of a property estimation workflow.
<i>WorkflowOutputToStore</i>	An object which describes which data should be cached after a workflow has finished executing, and from which completed protocols should the data be collected from.
<i>WorkflowSimulationDataToStore</i>	An object which describes which data should be cached after a workflow has finished executing, and from which completed protocols should the data be collected from.
<i>WorkflowDataCollectionToStore</i>	An object which describes which data should be cached after a workflow has finished executing, and from which completed protocols should the data be collected from.

WorkflowSchema

class propertyestimator.workflow.schemas.**WorkflowSchema** (*property_type=None*)

Outlines the workflow which should be followed when calculating a certain property.

__init__ (*property_type=None*)

Constructs a new WorkflowSchema object.

Parameters **property_type** (*str*) – The type of property which this workflow aims to estimate.

Methods

<code>__init__([property_type])</code>	Constructs a new WorkflowSchema object.
<code>json()</code>	Creates a JSON representation of this class.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.
<code>replace_protocol_types(protocol_replacements)</code>	Replaces protocols with given types with other protocols of specified replacements.
<code>validate_interfaces()</code>	Validates the flow of the data between protocols, ensuring that inputs and outputs correctly match up.

`replace_protocol_types` (*protocol_replacements*, *protocol_group_schema=None*)

Replaces protocols with given types with other protocols of specified replacements. This is useful when replacing the default protocols with custom ones, or swapping out base protocols with actual implementations

Warning: This method is NOT fully implemented and is likely to fail in all but a few specific cases. This method should be used with extreme caution.

Parameters

- **protocol_replacements** (*dict of str and str, None*) – A dictionary with keys of the types of protocols which should be replaced with those protocols named by the values.
- **protocol_group_schema** (*ProtocolGroupSchema*) – The protocol group to apply the replacements to. This is mainly used when applying this method recursively.

`validate_interfaces` ()

Validates the flow of the data between protocols, ensuring that inputs and outputs correctly match up.

`json` ()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

`classmethod parse_json` (*string_contents*, *encoding='utf8'*)

Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

ProtocolSchema

class `propertyestimator.workflow.schemas.ProtocolSchema`

A json serializable representation of a workflow protocol.

`__init__()`
Constructs a new ProtocolSchema object.

Methods

<code>__init__()</code>	Constructs a new ProtocolSchema object.
<code>json()</code>	Creates a JSON representation of this class.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.

`json()`
Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

classmethod `parse_json(string_contents, encoding='utf8')`
Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (`str` or `bytes`) – The typed json string.
- **encoding** (`str`) – The encoding of the `string_contents`.

Returns The parsed class.

Return type Any

ProtocolGroupSchema

class `propertyestimator.workflow.schemas.ProtocolGroupSchema`
A json serializable representation of a workflow protocol group.

`__init__()`
Constructs a new ProtocolGroupSchema object.

Methods

<code>__init__()</code>	Constructs a new ProtocolGroupSchema object.
<code>json()</code>	Creates a JSON representation of this class.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.

`json()`
Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

classmethod `parse_json(string_contents, encoding='utf8')`
Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

ProtocolReplicator

class propertyestimator.workflow.schemas.**ProtocolReplicator** (*replicator_id=""*)

A protocol replicator contains the information necessary to replicate parts of a property estimation workflow.

Any protocol whose id includes $\$(replicator.id)$ (where *replicator.id* is the id of a replicator) will be cloned for each value present in *template_values*. Protocols that are being replicated will also have any ReplicatorValue inputs replaced with the actual value taken from *template_values*.

When the protocol is replicated, the $\$(replicator.id)$ placeholder in the protocol id will be replaced an integer which corresponds to the index of a value in the *template_values* array.

Any protocols which take input from a replicated protocol will be updated to instead take a list of value, populated by the outputs of the replicated protocols.

Notes

- The *template_values* property must be a list of either constant values, or *ProtocolPath* objects which take their value from the *global* scope.
- If children of replicated protocols are also flagged as to be replicated, they will only have their ids changed to match the index of the parent protocol, as opposed to being fully replicated.

__init__ (*replicator_id=""*)

Constructs a new ProtocolReplicator object.

Parameters **replicator_id** (*str*) – The id of this replicator.

Methods

<code>__init__([replicator_id])</code>	Constructs a new ProtocolReplicator object.
<code>apply(protocols[, template_values, ...])</code>	Applies this replicator to the provided set of protocols and any of their children.
<code>json()</code>	Creates a JSON representation of this class.
<code>parse_json(string_contents[, encoding])</code>	Parses a typed json string into the corresponding class structure.
<code>update_references(protocols, ...)</code>	Redirects the input references of protocols to the replicated versions.

Attributes

<code>placeholder_id</code>	The string which protocols to be replicated should include in their ids.
-----------------------------	--

property_placeholder_id

The string which protocols to be replicated should include in their ids.

apply (*protocols, template_values=None, template_index=-1, template_value=None*)

Applies this replicator to the provided set of protocols and any of their children.

This protocol should be followed by a call to *update_references* to ensure that all protocols which take their input from a replicated protocol get correctly updated.

Parameters

- **protocols** (*dict of str and BaseProtocol*) – The protocols to apply the replicator to.

- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by this replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by this replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

Returns

- *dict of str and BaseProtocol* – The replicated protocols.
- *dict of ProtocolPath and list of tuple of ProtocolPath and int* – A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

update_references (*protocols, replication_map, template_values*)

Redirects the input references of protocols to the replicated versions.

Parameters

- **protocols** (*dict of str and BaseProtocol*) – The protocols which have had this replicator applied to them.

- **replication_map** (*dict of ProtocolPath and list of tuple of ProtocolPath and int*) – A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

- **template_values** (*List of Any*) – A list of the values which will be inserted into the newly replicated protocols.

json ()

Creates a JSON representation of this class.

Returns The JSON representation of this class.

Return type `str`

classmethod `parse_json` (*string_contents*, *encoding='utf8'*)
 Parses a typed json string into the corresponding class structure.

Parameters

- **string_contents** (*str or bytes*) – The typed json string.
- **encoding** (*str*) – The encoding of the *string_contents*.

Returns The parsed class.

Return type Any

WorkflowOutputToStore

class `propertyestimator.workflow.schemas.WorkflowOutputToStore`

An object which describes which data should be cached after a workflow has finished executing, and from which completed protocols should the data be collected from.

A *WorkflowOutputToStore* maps to the *BaseStoredData* stored data class.

substance

A reference to the composition of the collected data.

Type *ProtocolPath*

`__init__` ()

Constructs a new *WorkflowOutputToStore* object.

Methods

`__init__` ()

Constructs a new *WorkflowOutputToStore* object.

WorkflowSimulationDataToStore

class `propertyestimator.workflow.schemas.WorkflowSimulationDataToStore`

An object which describes which data should be cached after a workflow has finished executing, and from which completed protocols should the data be collected from.

A *WorkflowSimulationDataToStore* maps to the creation of a *StoredSimulationData* stored data class.

coordinate_file_path

A reference to the file path of a coordinate file which encodes the topology of the system.

Type *ProtocolPath*

trajectory_file_path

A reference to the file path of a .dcd trajectory file containing configurations generated by the simulation.

Type *ProtocolPath*

statistics_file_path

A reference to the file path of a *StatisticsArray* csv file, containing statistics generated by the simulation.

Type *ProtocolPath*

statistical_inefficiency

A reference to the statistical inefficiency of the collected data.

Type *ProtocolPath*

total_number_of_molecules

A reference to the total number of molecules in the system.

Type *ProtocolPath*

__init__()

Constructs a new WorkflowSimulationDataToStore object.

Methods

<code>__init__()</code>	Constructs a new WorkflowSimulationDataToStore object.
-------------------------	--

WorkflowDataCollectionToStore

class `propertyestimator.workflow.schemas.WorkflowDataCollectionToStore`

An object which describes which data should be cached after a workflow has finished executing, and from which completed protocols should the data be collected from.

A *WorkflowDataCollectionToStore* maps to the creation of a *StoredDataCollection* stored data class.

data

A dictionary of stored simulation data objects which have been given a unique key.

Type dict of str and WorkflowSimulationDataToStore

__init__()

Constructs a new WorkflowDataCollectionToStore object.

Methods

<code>__init__()</code>	Constructs a new WorkflowDataCollectionToStore object.
-------------------------	--

Base Protocol API

<i>BaseProtocol</i>	The base class for a protocol which would form one step of a larger property calculation workflow.
---------------------	--

BaseProtocol

class `propertyestimator.workflow.protocols.BaseProtocol` (*protocol_id*)

The base class for a protocol which would form one step of a larger property calculation workflow.

A protocol may for example:

- create the coordinates of a mixed simulation box
- set up a bound ligand-protein system
- build the simulation topology
- perform an energy minimisation

An individual protocol may require a set of inputs, which may either be set as constants

```
>>> from propertyestimator.protocols.simulation import RunOpenMMSimulation
>>>
>>> npt_equilibration = RunOpenMMSimulation('npt_equilibration')
>>> npt_equilibration.ensemble = RunOpenMMSimulation.Ensemble.NPT
```

or from the output of another protocol, pointed to by a ProtocolPath

```
>>> npt_production = RunOpenMMSimulation('npt_production')
>>> # Use the coordinate file output by the npt_equilibration protocol
>>> # as the input to the npt_production protocol
>>> npt_production.input_coordinate_file = ProtocolPath('output_coordinate_file',
>>>                                                    npt_equilibration.id)
```

In this way protocols may be chained together, thus defining a larger property calculation workflow from simple, reusable building blocks.

Warning: This class is still heavily under development and is subject to rapid changes.

`__init__(protocol_id)`
Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__(protocol_id)</code>	Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	tem- Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>id</code>	The unique id of this protocol.

Continued on next page

Table 97 – continued from previous page

<i>schema</i>	A serializable schema for this object.
property id	The unique id of this protocol.
Type	<code>str</code>
property schema	A serializable schema for this object.
Type	<code>ProtocolSchema</code>
property dependencies	A list of pointers to the protocols which this protocol takes input from.
Type	list of <code>ProtocolPath</code>
allow_merging	Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is <code>True</code> .
Type	<code>bool</code>
execute (<i>directory, available_resources</i>)	Execute the protocol.
	Protocols may be chained together by passing the output of previous protocols as input to the current one.
	Parameters
	<ul style="list-style-type: none"> • directory (<i>str</i>) – The directory to store output data in. • available_resources (<code>ComputeResources</code>) – The resources available to execute on.
	Returns The output of the execution.
	Return type <code>Dict[str, Any]</code>
set_uuid (<i>value</i>)	Store the uuid of the calculation this protocol belongs to
	Parameters value (<i>str</i>) – The uuid of the parent calculation.
replace_protocol (<i>old_id, new_id</i>)	Finds each input which came from a given protocol and redirects it to instead take input from a new one.
	Notes
	This method is mainly intended to be used only when merging multiple protocols into one.
	Parameters
	<ul style="list-style-type: none"> • old_id (<i>str</i>) – The id of the old input protocol. • new_id (<i>str</i>) – The id of the new input protocol.
can_merge (<i>other, path_replacements=None</i>)	Determines whether this protocol can be merged with another.
	Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type `bool`

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (*BaseProtocol*) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type `dict of ProtocolPath and ProtocolPath`

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input / output

Return type `Any`

set_value (*reference_path, value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format $\$(replicator.id)$).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

Input / Output Utilities

<i>PlaceholderInput</i>	A class to act as a place holder for a protocols input value, for when the value of an input is not known a priori, and does not come from another protocol.
<i>ReplicatorValue</i>	A placeholder value which will be set by a protocol replicator with the specified id.
<i>ProtocolPath</i>	Represents a pointer to the output of another protocol.

PlaceholderInput

class propertyestimator.workflow.utils.PlaceholderInput

A class to act as a place holder for a protocols input value, for when the value of an input is not known a priori, and does not come from another protocol.

__init__ ()
Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__</code>	Initialize self.
-----------------------	------------------

ReplicatorValue

class `propertyestimator.workflow.utils.ReplicatorValue` (*replicator_id=""*)

A placeholder value which will be set by a protocol replicator with the specified id.

`__init__` (*replicator_id=""*)

Constructs a new ReplicatorValue object

Parameters `replicator_id` (*str*) – The id of the replicator which will set this value.

Methods

<code>__init__</code> ([<i>replicator_id</i>])	Constructs a new ReplicatorValue object
--	---

ProtocolPath

class `propertyestimator.workflow.utils.ProtocolPath` (*property_name=""*, **protocol_ids*)

Represents a pointer to the output of another protocol.

`__init__` (*property_name=""*, **protocol_ids*)

Constructs a new ProtocolPath object.

Parameters

- **property_name** (*str*) – The property name referenced by the path.
- **protocol_ids** (*str*) – An args list of protocol ids in the order in which they will appear in the path.

Methods

<code>__init__</code> ([<i>property_name</i>])	Constructs a new ProtocolPath object.
<code>append_uuid</code> (<i>uuid</i>)	Appends a uuid to each of the protocol id's in the path
<code>from_string</code> (<i>existing_path_string</i>)	
<code>pop_next_in_path</code> ()	Pops and then returns the leading protocol id from the path.
<code>prepend_protocol_id</code> (<i>id_to_prepend</i>)	Prepend a new protocol id onto the front of the path.
<code>replace_protocol</code> (<i>old_id</i> , <i>new_id</i>)	Redirect the input to point at a new protocol.
<code>to_components</code> (<i>path_string</i>)	Splits a protocol path string into the property name, and the individual protocol ids.
<code>validate</code> (<i>v</i>)	

Attributes

<i>full_path</i>	The full path referenced by this object.
<i>is_global</i>	
<i>last_protocol</i>	The leading protocol id of the path.
<i>path_separator</i>	
<i>property_name</i>	The property name pointed to by the path.
<i>property_separator</i>	
<i>protocol_path</i>	The full path referenced by this object excluding the property name.
<i>start_protocol</i>	The leading protocol id of the path.

property property_name

The property name pointed to by the path.

Type str

property start_protocol

The leading protocol id of the path.

Type str

property last_protocol

The leading protocol id of the path.

Type str

property protocol_path

The full path referenced by this object excluding the property name.

Type str

property full_path

The full path referenced by this object.

Type str

static to_components (*path_string*)

Splits a protocol path string into the property name, and the individual protocol ids.

Parameters **path_string** (*str*) – The protocol path to split.

Returns A tuple of the property name, and a list of the protocol ids in the path.

Return type str, list of str

prepend_protocol_id (*id_to_prepend*)

Prepend a new protocol id onto the front of the path.

Parameters **id_to_prepend** (*str*) – The protocol id to prepend to the path

pop_next_in_path ()

Pops and then returns the leading protocol id from the path.

Returns The previously leading protocol id.

Return type str

append_uuid (*uuid*)

Appends a uuid to each of the protocol id's in the path

Parameters **uuid** (*str*) – The uuid to append.

replace_protocol (*old_id*, *new_id*)

Redirect the input to point at a new protocol.

The main use of this method is when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the protocol to replace.
- **new_id** (*str*) – The id of the new protocol to use.

Decorators

<i>protocol_input</i>	alias of <code>propertyestimator.workflow.decorators.ProtocolInputAttribute</code>
<i>protocol_output</i>	alias of <code>propertyestimator.workflow.decorators.ProtocolOutputAttribute</code>
<i>BaseProtocolAttribute</i>	A custom descriptor used to mark class attributes as being either a required input, or provided output of a protocol.
<i>MergeBehaviour</i>	A enum which describes how attributes should be handled when attempting to merge similar protocols.
<i>InequalityMergeBehaviour</i>	A enum which describes how attributes which can be compared with inequalities should be merged.

protocol_input

`propertyestimator.workflow.decorators.protocol_input`

alias of `propertyestimator.workflow.decorators.ProtocolInputAttribute`

protocol_output

`propertyestimator.workflow.decorators.protocol_output`

alias of `propertyestimator.workflow.decorators.ProtocolOutputAttribute`

BaseProtocolAttribute

class `propertyestimator.workflow.decorators.BaseProtocolAttribute` (*docstring*, *type_hint*)

A custom descriptor used to mark class attributes as being either a required input, or provided output of a protocol.

Notes

This decorator expects the protocol to have a matching private field in addition to the public attribute. For example if a protocol has an attribute *substance*, by default the protocol must also have a *_substance* field.

__init__ (*docstring*, *type_hint*)

Initializes a new `BaseProtocolAttribute` object.

Parameters

- **docstring** (*str*) – A docstring describing the attributes purpose. This will automatically be decorated with additional information such as type hints, default values, etc.

- **type_hint** (*type*, *typing.Union*) – The expected type of this attribute. This will be used to help the workflow engine ensure that expected input types match corresponding output values.

Methods

<code>__init__</code> (docstring, type_hint)	Initializes a new BaseProtocolAttribute object.
--	---

MergeBehaviour

class `propertyestimator.workflow.decorators.MergeBehaviour`

A enum which describes how attributes should be handled when attempting to merge similar protocols.

This enum may take values of

- `ExactlyEqual`: This attribute must be exactly equal between two protocols for them to be able to merge.

`__init__` ()

Initialize self. See `help(type(self))` for accurate signature.

Attributes

<code>ExactlyEqual</code>

InequalityMergeBehaviour

class `propertyestimator.workflow.decorators.InequalityMergeBehaviour`

A enum which describes how attributes which can be compared with inequalities should be merged.

This enum may take values of

- `SmallestValue`: When two protocols are merged, the smallest value of this attribute from either protocol is retained.
- `LargestValue`: When two protocols are merged, the largest value of this attribute from either protocol is retained.

`__init__` ()

Initialize self. See `help(type(self))` for accurate signature.

Attributes

<code>LargestValue</code>
<code>SmallestValue</code>

1.5.9 Built-in Workflow Protocols

Coordinate Generation

<i>BuildCoordinatesPackmol</i>	Creates a set of 3D coordinates with a specified composition using the PACKMOL package.
<i>SolvateExistingStructure</i>	Solvates a set of 3D coordinates with a specified solvent using the PACKMOL package.
<i>BuildDockedCoordinates</i>	Creates a set of coordinates for a ligand bound to some receptor.

BuildCoordinatesPackmol

class `propertyestimator.protocols.coordinates.BuildCoordinatesPackmol` (*protocol_id*)
Creates a set of 3D coordinates with a specified composition using the PACKMOL package.

`__init__` (*protocol_id*)
Constructs a new BuildCoordinatesPackmol object.

Methods

<code>__init__</code> (<i>protocol_id</i>)	Constructs a new BuildCoordinatesPackmol object.
<code>apply_replicator</code> (<i>replicator</i> , <i>template_values</i>)	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge</code> (<i>other</i> [, <i>path_replacements</i>])	Determines whether this protocol can be merged with another.
<code>execute</code> (<i>directory</i> , <i>available_resources</i>)	Execute the protocol.
<code>get_class_attribute</code> (<i>reference_path</i>)	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value</code> (<i>reference_path</i>)	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references</code> (<i>input_path</i>)	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge</code> (<i>other</i>)	Merges another BaseProtocol with this one.
<code>replace_protocol</code> (<i>old_id</i> , <i>new_id</i>)	Finds each input which came from a given protocol
<code>set_uuid</code> (<i>value</i>)	Store the uuid of the calculation this protocol belongs to
<code>set_value</code> (<i>reference_path</i> , <i>value</i>)	Sets the value of one of this protocols inputs.

Attributes

<i>allow_merging</i>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<i>box_aspect_ratio</i>	Protocol Input - The aspect ratio of the simulation box.
<i>coordinate_file_path</i>	Protocol Output - The file path to the created PDB coordinate file.
<i>dependencies</i>	A list of pointers to the protocols which this protocol takes input from.
<i>id</i>	The unique id of this protocol.
<i>mass_density</i>	Protocol Input - The target density of the created system.

Continued on next page

Table 109 – continued from previous page

<i>max_molecules</i>	Protocol Input - The maximum number of molecules to be added to the system.
<i>output_number_of_molecules</i>	Protocol Output - The number of molecules in the created system.
<i>output_substance</i>	Protocol Output - The substance which was built by packmol.
<i>retain_packmol_files</i>	Protocol Input - If True, packmol will not delete all of the temporary files it creates while building the coordinates.
<i>schema</i>	A serializable schema for this object.
<i>substance</i>	Protocol Input - The composition of the system to build.
<i>verbose_packmol</i>	Protocol Input - If True, packmol will print verbose information to the logger The default value of this attribute is <code>False</code> .

max_molecules

Protocol Input - The maximum number of molecules to be added to the system. The default value of this attribute is 1000.

Type `int`

mass_density

Protocol Input - The target density of the created system. The default value of this attribute is 0.95 gram / milliliter.

Type `Quantity`

box_aspect_ratio

Protocol Input - The aspect ratio of the simulation box. The default value of this attribute is [1.0, 1.0, 1.0].

Type `list`

substance

Protocol Input - The composition of the system to build. The default value of this attribute is not set and must be set by the user..

Type `Substance`

verbose_packmol

Protocol Input - If True, packmol will print verbose information to the logger The default value of this attribute is `False`.

Type `bool`

retain_packmol_files

Protocol Input - If True, packmol will not delete all of the temporary files it creates while building the coordinates. The default value of this attribute is `False`.

Type `bool`

output_number_of_molecules

Protocol Output - The number of molecules in the created system. This may be less than maximum requested due to rounding of mole fractions

Type `int`

output_substance

Protocol Output - The substance which was built by packmol. This may differ from the input substance

for system containing two or more components due to rounding of mole fractions. The mole fractions provided by this output should always be used when weighting values by a mole fraction.

Type *Substance*

coordinate_file_path

Protocol Output - The file path to the created PDB coordinate file.

Type *str*

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[*str*, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is *True*.

Type *bool*

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (BaseProtocol) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (ProtocolPath) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (ProtocolPath) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a *list / dict* which contains at least one ProtocolPath.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type str

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters *other* (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters *value* (*str*) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (`ProtocolPath`) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

SolvateExistingStructure

class `propertyestimator.protocols.coordinates.SolvateExistingStructure` (*protocol_id*)
Solvates a set of 3D coordinates with a specified solvent using the PACKMOL package.

__init__ (*protocol_id*)

Constructs a new SolvateExistingStructure object.

Methods

<code>__init__</code> (<i>protocol_id</i>)	Constructs a new SolvateExistingStructure object.
<code>apply_replicator</code> (<i>replicator</i> , <i>plate_values</i>)	Applies a <code>ProtocolReplicator</code> to this protocol.

Continued on next page

Table 110 – continued from previous page

<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>box_aspect_ratio</code>	Protocol Input - The aspect ratio of the simulation box.
<code>coordinate_file_path</code>	Protocol Output - The file path to the created PDB coordinate file.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>id</code>	The unique id of this protocol.
<code>mass_density</code>	Protocol Input - The target density of the created system.
<code>max_molecules</code>	Protocol Input - The maximum number of molecules to be added to the system.
<code>output_number_of_molecules</code>	Protocol Output - The number of molecules in the created system.
<code>output_substance</code>	Protocol Output - The substance which was built by packmol.
<code>retain_packmol_files</code>	Protocol Input - If True, packmol will not delete all of the temporary files it creates while building the coordinates.
<code>schema</code>	A serializable schema for this object.
<code>solute_coordinate_file</code>	Protocol Input - A file path to the solute to solvate.
<code>substance</code>	Protocol Input - The composition of the system to build.
<code>verbose_packmol</code>	Protocol Input - If True, packmol will print verbose information to the logger The default value of this attribute is <code>False</code> .

`solute_coordinate_file`

Protocol Input - A file path to the solute to solvate. The default value of this attribute is not set and must be set by the user..

Type `str`

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

box_aspect_ratio

Protocol Input - The aspect ratio of the simulation box. The default value of this attribute is [1.0, 1.0, 1.0].

Type list

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

coordinate_file_path

Protocol Output - The file path to the created PDB coordinate file.

Type str

property_dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a list / dict which contains at least one ProtocolPath.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type `str`

mass_density

Protocol Input - The target density of the created system. The default value of this attribute is 0.95 gram / milliliter.

Type `Quantity`

max_molecules

Protocol Input - The maximum number of molecules to be added to the system. The default value of this attribute is 1000.

Type `int`

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

output_number_of_molecules

Protocol Output - The number of molecules in the created system. This may be less than maximum requested due to rounding of mole fractions

Type `int`

output_substance

Protocol Output - The substance which was built by packmol. This may differ from the input substance for system containing two or more components due to rounding of mole fractions. The mole fractions provided by this output should always be used when weighting values by a mole fraction.

Type `Substance`

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (`str`) – The id of the old input protocol.
- **new_id** (`str`) – The id of the new input protocol.

retain_packmol_files

Protocol Input - If True, packmol will not delete all of the temporary files it creates while building the coordinates. The default value of this attribute is `False`.

Type `bool`

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path, value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

substance

Protocol Input - The composition of the system to build. The default value of this attribute is not set and must be set by the user..

Type *Substance*

verbose_packmol

Protocol Input - If True, packmol will print verbose information to the logger The default value of this attribute is False.

Type *bool*

BuildDockedCoordinates

class propertyestimator.protocols.coordinates.**BuildDockedCoordinates** (*protocol_id*)

Creates a set of coordinates for a ligand bound to some receptor.

Notes

This protocol currently only supports docking with the OpenEye OEDocking framework.

__init__ (*protocol_id*)

Constructs a new SolvateExistingStructure object.

Methods

<code>__init__(protocol_id)</code>	Constructs a new SolvateExistingStructure object.
<code>apply_replicator(replicator, plate_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.

Continued on next page

Table 112 – continued from previous page

<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>activate_site_location</code>	Protocol Input - Defines the method by which the activate site is identified.
<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>docked_complex_coordinate_path</code>	Protocol Output - The file path to the docked ligand-receptor complex.
<code>docked_ligand_coordinate_path</code>	Protocol Output - The file path to the coordinates of the ligand in it's docked pose, aligned with the initial <i>receptor_coordinate_file</i> .
<code>id</code>	The unique id of this protocol.
<code>ligand_residue_name</code>	Protocol Output - The residue name assigned to the docked ligand.
<code>ligand_substance</code>	Protocol Input - A substance containing only the ligand to dock.
<code>number_of_ligand_conformers</code>	Protocol Input - The number of conformers to try and dock into the receptor structure.
<code>receptor_coordinate_file</code>	Protocol Input - The file path to the MOL2 coordinates of the receptor molecule.
<code>receptor_residue_name</code>	Protocol Output - The residue name assigned to the receptor.
<code>schema</code>	A serializable schema for this object.

class ActivateSiteLocation

An enum which describes the methods by which a receptors activate site(s) is located.

ligand_substance

Protocol Input - A substance containing only the ligand to dock. The default value of this attribute is not set and must be set by the user..

Type *Substance*

number_of_ligand_conformers

Protocol Input - The number of conformers to try and dock into the receptor structure. The default value of this attribute is 100.

Type *int*

receptor_coordinate_file

Protocol Input - The file path to the MOL2 coordinates of the receptor molecule. The default value of this attribute is not set and must be set by the user..

Type `str`

activate_site_location

Protocol Input - Defines the method by which the activate site is identified. The default value of this attribute is `ActivateSiteLocation.ReceptorCenterOfMass`.

Type `BuildDockedCoordinates.ActivateSiteLocation`

docked_ligand_coordinate_path

Protocol Output - The file path to the coordinates of the ligand in it's docked pose, aligned with the initial `receptor_coordinate_file`.

Type `str`

docked_complex_coordinate_path

Protocol Output - The file path to the docked ligand-receptor complex.

Type `str`

ligand_residue_name

Protocol Output - The residue name assigned to the docked ligand.

Type `str`

receptor_residue_name

Protocol Output - The residue name assigned to the receptor.

Type `str`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator*, *template_values*, *template_index=-1*, *template_value=None*, *update_input_references=False*)

Applies a `ProtocolReplicator` to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format `$(replicator.id)`).

Parameters

- **replicator** (`ProtocolReplicator`) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with `template_index` and `template_value`

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with `template_values` and must be set along with a `template_value`.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with `template_values` and must be set along with a `template_index`.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the

actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a list / dict which contains at least one ProtocolPath.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (`str`) – The id of the old input protocol.
- **new_id** (`str`) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (`str`) – The uuid of the parent calculation.

set_value (*reference_path, value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (`ProtocolPath`) – The path pointing to the value to return.
- **value** (`Any`) – The value to set.

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (`str`) – The directory to store output data in.
- **available_resources** (`ComputeResources`) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

Force Field Assignment

<i>BuildSmirnoffSystem</i>	Parametrise a set of molecules with a given smirnoff force field using the OpenFF toolkit .
<i>BuildTLeapSystem</i>	Parametrise a set of molecules with an Amber based force field.

BuildSmirnoffSystem

class `propertyestimator.protocols.forcefield.BuildSmirnoffSystem(protocol_id)`
 Parametrise a set of molecules with a given smirnoff force field using the [OpenFF toolkit](#).

`__init__(protocol_id)`
 Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__(protocol_id)</code>	Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>apply_known_charges</code>	Protocol Input - If true, the formal charges of ions and the partial charges of the selected water model will be automatically applied to any matching molecules in the system.
<code>charged_molecule_paths</code>	Protocol Input - File paths to mol2 files which contain the charges assigned to molecules in the system.
<code>coordinate_file_path</code>	Protocol Input - The file path to the PDB coordinate file which defines the topology of the system to which the force field parameters will be assigned.

Continued on next page

Table 116 – continued from previous page

<i>dependencies</i>	A list of pointers to the protocols which this protocol takes input from.
<i>force_field_path</i>	Protocol Input - The file path to the force field parameters to assign to the system.
<i>id</i>	The unique id of this protocol.
<i>schema</i>	A serializable schema for this object.
<i>substance</i>	Protocol Input - The composition of the system.
<i>system_path</i>	Protocol Output - The path to the assigned system object.
<i>water_model</i>	Protocol Input - The water model to apply, if any water molecules are present.

charged_molecule_paths

Protocol Input - File paths to mol2 files which contain the charges assigned to molecules in the system. This input is helpful when dealing with large molecules (such as hosts in host-guest binding calculations) whose charges may be needed in multiple places, and hence should only be calculated once. The default value of this attribute is [].

Type list

apply_known_charges

Protocol Input - If true, the formal charges of ions and the partial charges of the selected water model will be automatically applied to any matching molecules in the system. The default value of this attribute is True.

Type bool

execute (*directory*, *available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

class WaterModel

An enum which describes which water model is being used, so that correct charges can be applied.

Warning: This is only a temporary addition until full water model support is introduced.

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

apply_replicator (*replicator*, *template_values*, *template_index=-1*, *template_value=None*, *update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format $\$(replicator.id)$).

Parameters

- **replicator** (`ProtocolReplicator`) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (`BaseProtocol`) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

coordinate_file_path

Protocol Input - The file path to the PDB coordinate file which defines the topology of the system to which the force field parameters will be assigned. The default value of this attribute is not set and must be set by the user..

Type str

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

force_field_path

Protocol Input - The file path to the force field parameters to assign to the system. The default value of this attribute is not set and must be set by the user..

Type `str`

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters `reference_path` (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters `reference_path` (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters `input_path` (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id (str)` – The id of the old input protocol.
- `new_id (str)` – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (value)

Store the uuid of the calculation this protocol belongs to

Parameters **value (str)** – The uuid of the parent calculation.

set_value (reference_path, value)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path (ProtocolPath)** – The path pointing to the value to return.
- **value (Any)** – The value to set.

substance

Protocol Input - The composition of the system. The default value of this attribute is not set and must be set by the user..

Type *Substance*

system_path

Protocol Output - The path to the assigned system object.

Type `str`

water_model

Protocol Input - The water model to apply, if any water molecules are present. The default value of this attribute is `WaterModel.TIP3P`.

Type `BaseBuildSystemProtocol.WaterModel`

BuildTLeapSystem

class `propertyestimator.protocols.forcefield.BuildTLeapSystem (protocol_id)`

Parametrise a set of molecules with an Amber based force field. using the `tleap` package.

Notes

- This protocol is currently a work in progress and as such has limited functionality compared to the more established *BuildSmirnoffSystem* protocol.
- This protocol requires the optional *ambertools ==19.0* dependency to be installed.

`__init__ (protocol_id)`

Initialize self. See `help(type(self))` for accurate signature.

Methods

`__init__ (protocol_id)`

Initialize self.

Continued on next page

Table 117 – continued from previous page

<code>apply_replicator(replicator, plate_values)</code>	tem-	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>		Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>		Execute the protocol.
<code>get_class_attribute(reference_path)</code>		Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>		Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>		Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>		Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>		Finds each input which came from a given protocol
<code>set_uuid(value)</code>		Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>		Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>charge_backend</code>	Protocol Input - The backend framework to use to assign partial charges.
<code>coordinate_file_path</code>	Protocol Input - The file path to the PDB coordinate file which defines the topology of the system to which the force field parameters will be assigned.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>force_field_path</code>	Protocol Input - The file path to the force field parameters to assign to the system.
<code>id</code>	The unique id of this protocol.
<code>schema</code>	A serializable schema for this object.
<code>substance</code>	Protocol Input - The composition of the system.
<code>system_path</code>	Protocol Output - The path to the assigned system object.
<code>water_model</code>	Protocol Input - The water model to apply, if any water molecules are present.

class ChargeBackend

The framework to use to assign partial charges.

charge_backend

Protocol Input - The backend framework to use to assign partial charges. The default value of this attribute is `ChargeBackend.OpenEye`.

Type *BuildTLeapSystem.ChargeBackend*

execute (directory, available_resources)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

class WaterModel

An enum which describes which water model is being used, so that correct charges can be applied.

Warning: This is only a temporary addition until full water model support is introduced.

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (`BaseProtocol`) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type `bool`

coordinate_file_path

Protocol Input - The file path to the PDB coordinate file which defines the topology of the system to which the force field parameters will be assigned. The default value of this attribute is not set and must be set by the user..

Type `str`

property_dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of `ProtocolPath`

force_field_path

Protocol Input - The file path to the force field parameters to assign to the system. The default value of this attribute is not set and must be set by the user..

Type `str`

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type *str*

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that can_merge has already returned that these protocols are compatible to be merged together.

Parameters **other** (*BaseProtocol*) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type Dict[*str*, *str*]

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

substance

Protocol Input - The composition of the system. The default value of this attribute is not set and must be set by the user..

Type *Substance*

system_path

Protocol Output - The path to the assigned system object.

Type *str*

water_model

Protocol Input - The water model to apply, if any water molecules are present. The default value of this attribute is `WaterModel.TIP3P`.

Type `BaseBuildSystemProtocol.WaterModel`

Simulation

<code>RunEnergyMinimisation</code>	A protocol to minimise the potential energy of a system.
<code>RunOpenMMSimulation</code>	Performs a molecular dynamics simulation in a given ensemble using an OpenMM backend.

RunEnergyMinimisation

class `propertyestimator.protocols.simulation.RunEnergyMinimisation` (*protocol_id*)
 A protocol to minimise the potential energy of a system.

`__init__` (*protocol_id*)
 Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__(protocol_id)</code>	Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>enable_pbc</code>	Protocol Input - If true, periodic boundary conditions will be enabled.
<code>id</code>	The unique id of this protocol.

Continued on next page

Table 121 – continued from previous page

<i>input_coordinate_file</i>	Protocol Input - The coordinates to minimise.
<i>max_iterations</i>	Protocol Input - The maximum number of iterations to perform.
<i>output_coordinate_file</i>	Protocol Output - The file path to the minimised coordinates.
<i>schema</i>	A serializable schema for this object.
<i>system_path</i>	Protocol Input - The path to the XML system object which defines the forces present in the system.
<i>tolerance</i>	Protocol Input - The energy tolerance to which the system should be minimized.

input_coordinate_file

Protocol Input - The coordinates to minimise. The default value of this attribute is not set and must be set by the user..

Type `str`

system_path

Protocol Input - The path to the XML system object which defines the forces present in the system. The default value of this attribute is not set and must be set by the user..

Type `str`

tolerance

Protocol Input - The energy tolerance to which the system should be minimized. The default value of this attribute is 10.0 kilojoule / mole.

Type Quantity

max_iterations

Protocol Input - The maximum number of iterations to perform. If this is 0, minimization is continued until the results converge without regard to how many iterations it takes. The default value of this attribute is 0.

Type `int`

enable_pbc

Protocol Input - If true, periodic boundary conditions will be enabled. The default value of this attribute is `True`.

Type `bool`

output_coordinate_file

Protocol Output - The file path to the minimised coordinates.

Type `str`

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type `bool`

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.

- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path, value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

RunOpenMMSimulation

class `propertyestimator.protocols.simulation.RunOpenMMSimulation` (*protocol_id*)

Performs a molecular dynamics simulation in a given ensemble using an OpenMM backend.

__init__ (*protocol_id*)

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__(protocol_id)</code>		Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	tem-	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>		Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>		Execute the protocol.
<code>get_class_attribute(reference_path)</code>		Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>		Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>		Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>		Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>		Finds each input which came from a given protocol
<code>set_uuid(value)</code>		Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>		Sets the value of one of this protocols inputs.

Attributes

<i>allow_gpu_platforms</i>	Protocol Input - If true, OpenMM will be allowed to run using a GPU if available, otherwise it will be constrained to only using CPUs.
<i>allow_merging</i>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<i>checkpoint_frequency</i>	Protocol Input - The frequency (in multiples of <i>output_frequency</i>) with which to write to a checkpoint file, e.g.
<i>dependencies</i>	A list of pointers to the protocols which this protocol takes input from.
<i>enable_pbc</i>	Protocol Input - If true, periodic boundary conditions will be enabled.
<i>ensemble</i>	Protocol Input - The thermodynamic ensemble to simulate in.
<i>high_precision</i>	Protocol Input - If true, OpenMM will be run using a platform with high precision settings.
<i>id</i>	The unique id of this protocol.
<i>input_coordinate_file</i>	Protocol Input - The file path to the starting coordinates.
<i>output_coordinate_file</i>	Protocol Output - The file path to the coordinates of the final system configuration.
<i>output_frequency</i>	Protocol Input - The frequency (in number of steps) with which to write to the output statistics and trajectory files.
<i>schema</i>	A serializable schema for this object.
<i>statistics_file_path</i>	Protocol Output - The file path to the statistics sampled during the simulation.
<i>steps_per_iteration</i>	Protocol Input - The number of steps to propogate the system by at each iteration.
<i>system_path</i>	Protocol Input - A path to the XML system object which defines the forces present in the system.
<i>thermodynamic_state</i>	Protocol Input - The thermodynamic conditions to simulate under The default value of this attribute is not set and must be set by the user..
<i>thermostat_friction</i>	Protocol Input - The thermostat friction coefficient.
<i>timestep</i>	Protocol Input - The timestep to evolve the system by at each step.
<i>total_number_of_iterations</i>	Protocol Input - The number of times to propogate the system forward by the <i>steps_per_iteration</i> number of steps.
<i>trajectory_file_path</i>	Protocol Output - The file path to the trajectory sampled during the simulation.

steps_per_iteration

Protocol Input - The number of steps to propogate the system by at each iteration. The total number of steps performed by this protocol will be *total_number_of_iterations* * *steps_per_iteration*. The default value of this attribute is 1000000.

Type int

total_number_of_iterations

Protocol Input - The number of times to propogate the system forward by the *steps_per_iteration* number of steps. The total number of steps performed by this protocol will be *total_number_of_iterations* *

steps_per_iteration. The default value of this attribute is 1.

Type `int`

output_frequency

Protocol Input - The frequency (in number of steps) with which to write to the output statistics and trajectory files. The default value of this attribute is 3000. When two protocols are merged, the largest value of this attribute from either protocol is retained.

Type `int`

checkpoint_frequency

Protocol Input - The frequency (in multiples of *output_frequency*) with which to write to a checkpoint file, e.g. if *output_frequency*=100 and *checkpoint_frequency*=2, a checkpoint file would be saved every 200 steps. The default value of this attribute is 10. When two protocols are merged, the largest value of this attribute from either protocol is retained. This input is *optional*.

Type `int`

timestep

Protocol Input - The timestep to evolve the system by at each step. The default value of this attribute is 2.0 *femtosecond*. When two protocols are merged, the largest value of this attribute from either protocol is retained.

Type `Quantity`

thermodynamic_state

Protocol Input - The thermodynamic conditions to simulate under The default value of this attribute is not set and must be set by the user..

Type `ThermodynamicState`

ensemble

Protocol Input - The thermodynamic ensemble to simulate in. The default value of this attribute is `Ensemble.NPT`.

Type `Ensemble`

thermostat_friction

Protocol Input - The thermostat friction coefficient. The default value of this attribute is 1.0 / *picosecond*. When two protocols are merged, the largest value of this attribute from either protocol is retained.

Type `Quantity`

input_coordinate_file

Protocol Input - The file path to the starting coordinates. The default value of this attribute is not set and must be set by the user..

Type `str`

system_path

Protocol Input - A path to the XML system object which defines the forces present in the system. The default value of this attribute is not set and must be set by the user..

Type `str`

enable_pbc

Protocol Input - If true, periodic boundary conditions will be enabled. The default value of this attribute is `True`.

Type `bool`

allow_gpu_platforms

Protocol Input - If true, OpenMM will be allowed to run using a GPU if available, otherwise it will be constrained to only using CPUs. The default value of this attribute is `True`.

Type `bool`

high_precision

Protocol Input - If true, OpenMM will be run using a platform with high precision settings. This will be the Reference platform when only a CPU is available, or double precision mode when a GPU is available. The default value of this attribute is `False`.

Type `bool`

output_coordinate_file

Protocol Output - The file path to the coordinates of the final system configuration.

Type `str`

trajectory_file_path

Protocol Output - The file path to the trajectory sampled during the simulation.

Type `str`

statistics_file_path

Protocol Output - The file path to the statistics sampled during the simulation.

Type `str`

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters *input_path* (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters *other* (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters *value* (*str*) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

- **value** (*Any*) – The value to set.

YANK Free Energies

<i>BaseYankProtocol</i>	An abstract base class for protocols which will performs a set of alchemical free energy simulations using the YANK framework.
<i>LigandReceptorYankProtocol</i>	A protocol for performing ligand-receptor alchemical free energy calculations using the YANK framework.
<i>SolvationYankProtocol</i>	A protocol for performing solvation alchemical free energy calculations using the YANK framework.

BaseYankProtocol

class propertyestimator.protocols.yank.**BaseYankProtocol** (*protocol_id*)

An abstract base class for protocols which will performs a set of alchemical free energy simulations using the YANK framework.

Protocols which inherit from this base must implement the abstract `_get_yank_options` methods.

`__init__` (*protocol_id*)

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__(protocol_id)</code>	Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>checkpoint_interval</code>	Protocol Input - The number of iterations between saving YANK checkpoint files.

Continued on next page

Table 126 – continued from previous page

<i>dependencies</i>	A list of pointers to the protocols which this protocol takes input from.
<i>estimated_free_energy</i>	Protocol Output - The estimated free energy value and its uncertainty returned by YANK.
<i>id</i>	The unique id of this protocol.
<i>number_of_equilibration_iterations</i>	Protocol Input - The number of iterations used for equilibration before production run.
<i>number_of_iterations</i>	Protocol Input - The number of YANK iterations to perform.
<i>schema</i>	A serializable schema for this object.
<i>setup_only</i>	Protocol Input - If true, YANK will only create and validate the setup files, but not actually run any simulations.
<i>steps_per_iteration</i>	Protocol Input - The number of steps per YANK iteration to perform.
<i>thermodynamic_state</i>	Protocol Input - The state at which to run the calculations.
<i>timestep</i>	Protocol Input - The length of the timestep to take.
<i>verbose</i>	Protocol Input - Controls whether or not to run YANK at high verbosity.

thermodynamic_state

Protocol Input - The state at which to run the calculations. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

number_of_equilibration_iterations

Protocol Input - The number of iterations used for equilibration before production run. Only post-equilibration iterations are written to file. The default value of this attribute is 1.

Type *int*

number_of_iterations

Protocol Input - The number of YANK iterations to perform. The default value of this attribute is 5000.

Type *int*

steps_per_iteration

Protocol Input - The number of steps per YANK iteration to perform. The default value of this attribute is 500.

Type *int*

checkpoint_interval

Protocol Input - The number of iterations between saving YANK checkpoint files. The default value of this attribute is 50. When two protocols are merged, the largest value of this attribute from either protocol is retained.

Type *int*

timestep

Protocol Input - The length of the timestep to take. The default value of this attribute is 2 femtosecond. When two protocols are merged, the largest value of this attribute from either protocol is retained.

Type *Quantity*

verbose

Protocol Input - Controls whether or not to run YANK at high verbosity. The default value of this attribute is `False`.

Type `bool`

setup_only

Protocol Input - If true, YANK will only create and validate the setup files, but not actually run any simulations. This argument is mainly only to be used for testing purposes. The default value of this attribute is `False`.

Type `bool`

estimated_free_energy

Protocol Output - The estimated free energy value and its uncertainty returned by YANK.

Type `EstimatedQuantity`

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (`ComputeResources`) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a `ProtocolReplicator` to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format `$(replicator.id)`).

Parameters

- **replicator** (`ProtocolReplicator`) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with `template_index` and `template_value`
- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with `template_values` and must be set along with a `template_value`.
- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type *bool*

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type *object*

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a *list / dict* which contains at least one ProtocolPath.

Parameters `input_path` (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by `input_path` depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type Dict[`str`, `str`]

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id` (`str`) – The id of the old input protocol.
- `new_id` (`str`) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters `value` (`str`) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- `reference_path` (`ProtocolPath`) – The path pointing to the value to return.
- `value` (`Any`) – The value to set.

LigandReceptorYankProtocol

class `propertyestimator.protocols.yank.LigandReceptorYankProtocol` (*protocol_id*)

A protocol for performing ligand-receptor alchemical free energy calculations using the YANK framework.

`__init__(protocol_id)`
Constructs a new LigandReceptorYankProtocol object.

Methods

<code>__init__(protocol_id)</code>	Constructs a new LigandReceptorYankProtocol object.
<code>apply_replicator(replicator, plate_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>apply_restraints</code>	Protocol Input - Determines whether the ligand should be explicitly restrained to the receptor in order to stop the ligand from temporarily unbinding.
<code>checkpoint_interval</code>	Protocol Input - The number of iterations between saving YANK checkpoint files.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>estimated_free_energy</code>	Protocol Output - The estimated free energy value and its uncertainty returned by YANK.
<code>force_field_path</code>	Protocol Input - The path to the force field which defines the charge method to use for the calculation.
<code>id</code>	The unique id of this protocol.
<code>ligand_residue_name</code>	Protocol Input - The residue name of the ligand.
<code>number_of_equilibration_iterations</code>	Protocol Input - The number of iterations used for equilibration before production run.
<code>number_of_iterations</code>	Protocol Input - The number of YANK iterations to perform.
<code>receptor_residue_name</code>	Protocol Input - The residue name of the receptor.

Continued on next page

Table 128 – continued from previous page

<i>restraint_type</i>	Protocol Input - The type of ligand restraint applied, provided that <i>apply_restraints</i> is <i>True</i> . The default value of this attribute is <code>RestraintType.Harmonic</code> .
<i>schema</i>	A serializable schema for this object.
<i>setup_only</i>	Protocol Input - If true, YANK will only create and validate the setup files, but not actually run any simulations.
<i>solvated_complex_coordinates</i>	Protocol Input - The file path to the solvated complex coordinates.
<i>solvated_complex_system</i>	Protocol Input - The file path to the solvated complex system object.
<i>solvated_complex_trajectory_path</i>	Protocol Output - The file path to the generated ligand trajectory.
<i>solvated_ligand_coordinates</i>	Protocol Input - The file path to the solvated ligand coordinates.
<i>solvated_ligand_system</i>	Protocol Input - The file path to the solvated ligand system object.
<i>solvated_ligand_trajectory_path</i>	Protocol Output - The file path to the generated ligand trajectory.
<i>steps_per_iteration</i>	Protocol Input - The number of steps per YANK iteration to perform.
<i>thermodynamic_state</i>	Protocol Input - The state at which to run the calculations.
<i>timestep</i>	Protocol Input - The length of the timestep to take.
<i>verbose</i>	Protocol Input - Controls whether or not to run YANK at high verbosity.

class RestraintType

The types of ligand restraints available within yank.

ligand_residue_name

Protocol Input - The residue name of the ligand. The default value of this attribute is not set and must be set by the user..

Type `str`

receptor_residue_name

Protocol Input - The residue name of the receptor. The default value of this attribute is not set and must be set by the user..

Type `str`

solvated_ligand_coordinates

Protocol Input - The file path to the solvated ligand coordinates. The default value of this attribute is not set and must be set by the user..

Type `str`

solvated_ligand_system

Protocol Input - The file path to the solvated ligand system object. The default value of this attribute is not set and must be set by the user..

Type `str`

solvated_complex_coordinates

Protocol Input - The file path to the solvated complex coordinates. The default value of this attribute is

not set and must be set by the user..

Type `str`

solvated_complex_system

Protocol Input - The file path to the solvated complex system object. The default value of this attribute is not set and must be set by the user..

Type `str`

force_field_path

Protocol Input - The path to the force field which defines the charge method to use for the calculation. The default value of this attribute is not set and must be set by the user..

Type `str`

apply_restraints

Protocol Input - Determines whether the ligand should be explicitly restrained to the receptor in order to stop the ligand from temporarily unbinding. The default value of this attribute is `True`.

Type `bool`

restraint_type

Protocol Input - The type of ligand restraint applied, provided that `apply_restraints` is `True` The default value of this attribute is `RestraintType.Harmonic`.

Type `LigandReceptorYankProtocol.RestraintType`

solvated_ligand_trajectory_path

Protocol Output - The file path to the generated ligand trajectory.

Type `str`

solvated_complex_trajectory_path

Protocol Output - The file path to the generated ligand trajectory.

Type `str`

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (`str`) – The directory to store output data in.
- **available_resources** (`ComputeResources`) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a `ProtocolReplicator` to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format `$(replicator.id)`).

Parameters

- **replicator** (`ProtocolReplicator`) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (`BaseProtocol`) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

checkpoint_interval

Protocol Input - The number of iterations between saving YANK checkpoint files. The default value of this attribute is 50. When two protocols are merged, the largest value of this attribute from either protocol is retained.

Type int

property_dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

estimated_free_energy

Protocol Output - The estimated free energy value and its uncertainty returned by YANK.

Type EstimatedQuantity

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

number_of_equilibration_iterations

Protocol Input - The number of iterations used for equilibration before production run. Only post-equilibration iterations are written to file. The default value of this attribute is 1.

Type `int`

number_of_iterations

Protocol Input - The number of YANK iterations to perform. The default value of this attribute is 5000.

Type `int`

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

setup_only

Protocol Input - If true, YANK will only create and validate the setup files, but not actually run any simulations. This argument is mainly only to be used for testing purposes. The default value of this attribute is `False`.

Type `bool`

steps_per_iteration

Protocol Input - The number of steps per YANK iteration to perform. The default value of this attribute is 500.

Type `int`

thermodynamic_state

Protocol Input - The state at which to run the calculations. The default value of this attribute is not set and must be set by the user..

Type *ThermodynamicState*

timestep

Protocol Input - The length of the timestep to take. The default value of this attribute is 2 femtosecond. When two protocols are merged, the largest value of this attribute from either protocol is retained.

Type `Quantity`

verbose

Protocol Input - Controls whether or not to run YANK at high verbosity. The default value of this attribute is `False`.

Type `bool`

SolvationYankProtocol

class propertyestimator.protocols.yank.**SolvationYankProtocol** (*protocol_id*)

A protocol for performing solvation alchemical free energy calculations using the YANK framework.

This protocol can be used for box solvation free energies (setting the *solvent_1* input to the solvent of interest and setting *solvent_2* as an empty *Substance*) or transfer free energies (setting both the *solvent_1* and *solvent_2* inputs to different solvents).

`__init__` (*protocol_id*)

Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__</code> (<i>protocol_id</i>)	Initialize self.
<code>apply_replicator</code> (<i>replicator</i> , <i>template_values</i>)	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge</code> (<i>other</i> [, <i>path_replacements</i>])	Determines whether this protocol can be merged with another.
<code>execute</code> (<i>directory</i> , <i>available_resources</i>)	Execute the protocol.
<code>get_class_attribute</code> (<i>reference_path</i>)	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value</code> (<i>reference_path</i>)	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references</code> (<i>input_path</i>)	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge</code> (<i>other</i>)	Merges another BaseProtocol with this one.
<code>replace_protocol</code> (<i>old_id</i> , <i>new_id</i>)	Finds each input which came from a given protocol
<code>set_uuid</code> (<i>value</i>)	Store the uuid of the calculation this protocol belongs to
<code>set_value</code> (<i>reference_path</i> , <i>value</i>)	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>checkpoint_interval</code>	Protocol Input - The number of iterations between saving YANK checkpoint files.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>electrostatic_lambdas_1</code>	Protocol Input - The list of electrostatic alchemical states that YANK should sample at.
<code>electrostatic_lambdas_2</code>	Protocol Input - The list of electrostatic alchemical states that YANK should sample at.
<code>estimated_free_energy</code>	Protocol Output - The estimated free energy value and its uncertainty returned by YANK.
<code>id</code>	The unique id of this protocol.
<code>number_of_equilibration_iterations</code>	Protocol Input - The number of iterations used for equilibration before production run.

Continued on next page

Table 130 – continued from previous page

<i>number_of_iterations</i>	Protocol Input - The number of YANK iterations to perform.
<i>schema</i>	A serializable schema for this object.
<i>setup_only</i>	Protocol Input - If true, YANK will only create and validate the setup files, but not actually run any simulations.
<i>solute</i>	Protocol Input - The substance describing the composition of the solute.
<i>solvent_1</i>	Protocol Input - The substance describing the composition of the first solvent.
<i>solvent_1_coordinates</i>	Protocol Input - The file path to the coordinates of the solute embedded in the first solvent.
<i>solvent_1_system</i>	Protocol Input - The file path to the system object of the solute embedded in the first solvent.
<i>solvent_1_trajectory_path</i>	Protocol Output - The file path to the trajectory of the solute in the first solvent.
<i>solvent_2</i>	Protocol Input - The substance describing the composition of the second solvent.
<i>solvent_2_coordinates</i>	Protocol Input - The file path to the coordinates of the solute embedded in the second solvent.
<i>solvent_2_system</i>	Protocol Input - The file path to the system object of the solute embedded in the second solvent.
<i>solvent_2_trajectory_path</i>	Protocol Output - The file path to the trajectory of the solute in the second solvent.
<i>steps_per_iteration</i>	Protocol Input - The number of steps per YANK iteration to perform.
<i>steric_lambdas_1</i>	Protocol Input - The list of steric alchemical states that YANK should sample at.
<i>steric_lambdas_2</i>	Protocol Input - The list of steric alchemical states that YANK should sample at.
<i>thermodynamic_state</i>	Protocol Input - The state at which to run the calculations.
<i>timestep</i>	Protocol Input - The length of the timestep to take.
<i>verbose</i>	Protocol Input - Controls whether or not to run YANK at high verbosity.

solute

Protocol Input - The substance describing the composition of the solute. This should include the solute molecule as well as any counter ions. The default value of this attribute is not set and must be set by the user..

Type *Substance*

solvent_1

Protocol Input - The substance describing the composition of the first solvent. The default value of this attribute is not set and must be set by the user..

Type *Substance*

solvent_2

Protocol Input - The substance describing the composition of the second solvent. The default value of this attribute is not set and must be set by the user..

Type *Substance*

solvent_1_coordinates

Protocol Input - The file path to the coordinates of the solute embedded in the first solvent. The default value of this attribute is not set and must be set by the user.

Type `str`

solvent_1_system

Protocol Input - The file path to the system object of the solute embedded in the first solvent. The default value of this attribute is not set and must be set by the user.

Type `str`

solvent_2_coordinates

Protocol Input - The file path to the coordinates of the solute embedded in the second solvent. The default value of this attribute is not set and must be set by the user.

Type `str`

solvent_2_system

Protocol Input - The file path to the system object of the solute embedded in the second solvent. The default value of this attribute is not set and must be set by the user.

Type `str`

electrostatic_lambdas_1

Protocol Input - The list of electrostatic alchemical states that YANK should sample at. These values will be passed to the YANK `lambda_electrostatics` option. If no option is set, YANK will use `trailblaze` algorithm to determine this option automatically. The default value of this attribute is not set. This input is *optional*.

Type `list`

steric_lambdas_1

Protocol Input - The list of steric alchemical states that YANK should sample at. These values will be passed to the YANK `lambda_sterics` option. If no option is set, YANK will use `trailblaze` algorithm to determine this option automatically. The default value of this attribute is not set. This input is *optional*.

Type `list`

electrostatic_lambdas_2

Protocol Input - The list of electrostatic alchemical states that YANK should sample at. These values will be passed to the YANK `lambda_electrostatics` option. If no option is set, YANK will use `trailblaze` algorithm to determine this option automatically. The default value of this attribute is not set. This input is *optional*.

Type `list`

steric_lambdas_2

Protocol Input - The list of steric alchemical states that YANK should sample at. These values will be passed to the YANK `lambda_sterics` option. If no option is set, YANK will use `trailblaze` algorithm to determine this option automatically. The default value of this attribute is not set. This input is *optional*.

Type `list`

solvent_1_trajectory_path

Protocol Output - The file path to the trajectory of the solute in the first solvent.

Type `str`

solvent_2_trajectory_path

Protocol Output - The file path to the trajectory of the solute in the second solvent.

Type `str`

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (`BaseProtocol`) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type `bool`

checkpoint_interval

Protocol Input - The number of iterations between saving YANK checkpoint files. The default value of this attribute is 50. When two protocols are merged, the largest value of this attribute from either protocol is retained.

Type `int`

property_dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of `ProtocolPath`

estimated_free_energy

Protocol Output - The estimated free energy value and its uncertainty returned by YANK.

Type `EstimatedQuantity`

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

number_of_equilibration_iterations

Protocol Input - The number of iterations used for equilibration before production run. Only post-equilibration iterations are written to file. The default value of this attribute is 1.

Type `int`

number_of_iterations

Protocol Input - The number of YANK iterations to perform. The default value of this attribute is 5000.

Type `int`

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id` (*str*) – The id of the old input protocol.
- `new_id` (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters `value` (*str*) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- `reference_path` (`ProtocolPath`) – The path pointing to the value to return.
- `value` (*Any*) – The value to set.

setup_only

Protocol Input - If true, YANK will only create and validate the setup files, but not actually run any simulations. This argument is mainly only to be used for testing purposes. The default value of this attribute is `False`.

Type `bool`

steps_per_iteration

Protocol Input - The number of steps per YANK iteration to perform. The default value of this attribute is 500.

Type `int`

thermodynamic_state

Protocol Input - The state at which to run the calculations. The default value of this attribute is not set and must be set by the user.

Type `ThermodynamicState`

timestep

Protocol Input - The length of the timestep to take. The default value of this attribute is 2 femtosecond. When two protocols are merged, the largest value of this attribute from either protocol is retained.

Type `Quantity`

verbose

Protocol Input - Controls whether or not to run YANK at high verbosity. The default value of this attribute is `False`.

Type `bool`

Simulation Analysis

<i>AveragePropertyProtocol</i>	An abstract base class for protocols which will calculate the average of a property and its uncertainty via bootstrapping.
<i>AverageTrajectoryProperty</i>	An abstract base class for protocols which will calculate the average of a property from a simulation trajectory.
<i>ExtractAverageStatistic</i>	Extracts the average value from a statistics file which was generated during a simulation.
<i>ExtractUncorrelatedData</i>	An abstract base class for protocols which will subsample a data set, yielding only equilibrated, uncorrelated data.
<i>ExtractUncorrelatedTrajectoryData</i>	A protocol which will subsample frames from a trajectory, yielding only uncorrelated frames as determined from a provided statistical inefficiency and equilibration time.
<i>ExtractUncorrelatedStatisticsData</i>	A protocol which will subsample entries from a statistics array, yielding only uncorrelated entries as determined from a provided statistical inefficiency and equilibration time.

AveragePropertyProtocol

class `propertyestimator.protocols.analysis.AveragePropertyProtocol` (*protocol_id*)

An abstract base class for protocols which will calculate the average of a property and its uncertainty via bootstrapping.

__init__ (*protocol_id*)

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__(protocol_id)</code>		Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	tem-	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>		Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>		Execute the protocol.
<code>get_class_attribute(reference_path)</code>		Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>		Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>		Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>		Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>		Finds each input which came from a given protocol
<code>set_uuid(value)</code>		Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>		Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>bootstrap_iterations</code>	Protocol Input - The number of bootstrap iterations to perform.
<code>bootstrap_sample_size</code>	Protocol Input - The relative sample size to use for bootstrapping.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>equilibration_index</code>	Protocol Output - The index in the data set after which the data is stationary.
<code>id</code>	The unique id of this protocol.
<code>schema</code>	A serializable schema for this object.
<code>statistical_inefficiency</code>	Protocol Output - The statistical inefficiency in the data set.
<code>uncorrelated_values</code>	Protocol Output - The uncorrelated values which the average was calculated from.
<code>value</code>	Protocol Output - The average value and its uncertainty.

bootstrap_iterations

Protocol Input - The number of bootstrap iterations to perform. The default value of this attribute is 250.

Type int

bootstrap_sample_size

Protocol Input - The relative sample size to use for bootstrapping. The default value of this attribute is 1.0.

Type float

equilibration_index

Protocol Output - The index in the data set after which the data is stationary.

Type `int`

statistical_inefficiency

Protocol Output - The statistical inefficiency in the data set.

Type `float`

value

Protocol Output - The average value and its uncertainty.

Type `EstimatedQuantity`

uncorrelated_values

Protocol Output - The uncorrelated values which the average was calculated from.

Type `Quantity`

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (`ComputeResources`) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (`ProtocolReplicator`) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type *bool*

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type *object*

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a *list / dict* which contains at least one ProtocolPath.

Parameters `input_path` (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by `input_path` depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type Dict[`str`, `str`]

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id` (`str`) – The id of the old input protocol.
- `new_id` (`str`) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters `value` (`str`) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- `reference_path` (`ProtocolPath`) – The path pointing to the value to return.
- `value` (`Any`) – The value to set.

AverageTrajectoryProperty

class `propertyestimator.protocols.analysis.AverageTrajectoryProperty` (*protocol_id*)

An abstract base class for protocols which will calculate the average of a property from a simulation trajectory.

`__init__(protocol_id)`
 Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__(protocol_id)</code>	Initialize self.
<code>apply_replicator(replicator, template_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>bootstrap_iterations</code>	Protocol Input - The number of bootstrap iterations to perform.
<code>bootstrap_sample_size</code>	Protocol Input - The relative sample size to use for bootstrapping.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>equilibration_index</code>	Protocol Output - The index in the data set after which the data is stationary.
<code>id</code>	The unique id of this protocol.
<code>input_coordinate_file</code>	Protocol Input - The file path to the starting coordinates of a trajectory.
<code>schema</code>	A serializable schema for this object.
<code>statistical_inefficiency</code>	Protocol Output - The statistical inefficiency in the data set.
<code>trajectory_path</code>	Protocol Input - The file path to the trajectory to average over.
<code>uncorrelated_values</code>	Protocol Output - The uncorrelated values which the average was calculated from.
<code>value</code>	Protocol Output - The average value and its uncertainty.

input_coordinate_file

Protocol Input - The file path to the starting coordinates of a trajectory. The default value of this attribute is not set and must be set by the user..

Type `str`

trajectory_path

Protocol Input - The file path to the trajectory to average over. The default value of this attribute is not set and must be set by the user..

Type `str`

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (`ComputeResources`) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a `ProtocolReplicator` to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format `$(replicator.id)`).

Parameters

- **replicator** (`ProtocolReplicator`) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with `template_index` and `template_value`

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with `template_values` and must be set along with a `template_value`.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with `template_values` and must be set along with a `template_index`.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the

actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

bootstrap_iterations

Protocol Input - The number of bootstrap iterations to perform. The default value of this attribute is 250.

Type int

bootstrap_sample_size

Protocol Input - The relative sample size to use for bootstrapping. The default value of this attribute is 1.0.

Type float

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (BaseProtocol) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property_dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

equilibration_index

Protocol Output - The index in the data set after which the data is stationary.

Type int

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (ProtocolPath) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (ProtocolPath) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters `input_path` (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id` (*str*) – The id of the old input protocol.
- `new_id` (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters `value` (*str*) – The uuid of the parent calculation.

set_value (*reference_path, value*)

Sets the value of one of this protocols inputs.

Parameters

- `reference_path` (`ProtocolPath`) – The path pointing to the value to return.

- **value** (*Any*) – The value to set.

statistical_inefficiency

Protocol Output - The statistical inefficiency in the data set.

Type float

uncorrelated_values

Protocol Output - The uncorrelated values which the average was calculated from.

Type Quantity

value

Protocol Output - The average value and its uncertainty.

Type EstimatedQuantity

ExtractAverageStatistic

class propertyestimator.protocols.analysis.**ExtractAverageStatistic** (*protocol_id*)

Extracts the average value from a statistics file which was generated during a simulation.

__init__ (*protocol_id*)

Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__(protocol_id)</code>	Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	tem- Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>bootstrap_iterations</code>	Protocol Input - The number of bootstrap iterations to perform.

Continued on next page

Table 137 – continued from previous page

<i>bootstrap_sample_size</i>	Protocol Input - The relative sample size to use for bootstrapping.
<i>dependencies</i>	A list of pointers to the protocols which this protocol takes input from.
<i>divisor</i>	Protocol Input - A value to divide the statistic by.
<i>equilibration_index</i>	Protocol Output - The index in the data set after which the data is stationary.
<i>id</i>	The unique id of this protocol.
<i>schema</i>	A serializable schema for this object.
<i>statistical_inefficiency</i>	Protocol Output - The statistical inefficiency in the data set.
<i>statistics_path</i>	Protocol Input - The file path to the statistics to average over.
<i>statistics_type</i>	Protocol Input - The type of statistic to average over.
<i>uncorrelated_values</i>	Protocol Output - The uncorrelated values which the average was calculated from.
<i>value</i>	Protocol Output - The average value and its uncertainty.

statistics_path

Protocol Input - The file path to the statistics to average over. The default value of this attribute is not set and must be set by the user..

Type `str`

statistics_type

Protocol Input - The type of statistic to average over. The default value of this attribute is not set and must be set by the user..

Type `ObservableType`

divisor

Protocol Input - A value to divide the statistic by. This is useful if a statistic (such as enthalpy) needs to be normalised by the number of molecules. The default value of this attribute is `1.0`.

Type `typing.Union[int, float, pint.quantity.build_quantity_class.<locals>.Quantity]`

execute (*directory*, *available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (`ComputeResources`) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format $\$(replicator.id)$).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

bootstrap_iterations

Protocol Input - The number of bootstrap iterations to perform. The default value of this attribute is 250.

Type int

bootstrap_sample_size

Protocol Input - The relative sample size to use for bootstrapping. The default value of this attribute is 1.0.

Type float

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

equilibration_index

Protocol Output - The index in the data set after which the data is stationary.

Type int

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (ProtocolPath) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (ProtocolPath) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a *list / dict* which contains at least one ProtocolPath.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type str

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (BaseProtocol) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type Dict[str, str]

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

statistical_inefficiency

Protocol Output - The statistical inefficiency in the data set.

Type *float*

uncorrelated_values

Protocol Output - The uncorrelated values which the average was calculated from.

Type *Quantity*

value

Protocol Output - The average value and its uncertainty.

Type *EstimatedQuantity*

ExtractUncorrelatedData

class `propertyestimator.protocols.analysis.ExtractUncorrelatedData` (*protocol_id*)

An abstract base class for protocols which will subsample a data set, yielding only equilibrated, uncorrelated data.

__init__ (*protocol_id*)

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__</code> (<i>protocol_id</i>)	Initialize self.
--	------------------

Continued on next page

Table 138 – continued from previous page

<code>apply_replicator(replicator, plate_values)</code>	tem-	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>		Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>		Execute the protocol.
<code>get_class_attribute(reference_path)</code>		Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>		Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>		Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>		Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>		Finds each input which came from a given protocol
<code>set_uuid(value)</code>		Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>		Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>equilibration_index</code>	Protocol Input - The index in the data set after which the data is stationary.
<code>id</code>	The unique id of this protocol.
<code>number_of_uncorrelated_samples</code>	Protocol Output - The number of uncorrelated samples.
<code>schema</code>	A serializable schema for this object.
<code>statistical_inefficiency</code>	Protocol Input - The statistical inefficiency in the data set.

equilibration_index

Protocol Input - The index in the data set after which the data is stationary. The default value of this attribute is not set and must be set by the user..

Type int

statistical_inefficiency

Protocol Input - The statistical inefficiency in the data set. The default value of this attribute is not set and must be set by the user..

Type float

number_of_uncorrelated_samples

Protocol Output - The number of uncorrelated samples.

Type int

execute (directory, available_resources)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.

- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type `bool`

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (ProtocolPath) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (ProtocolPath) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a *list / dict* which contains at least one ProtocolPath.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (BaseProtocol) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type Dict[str, str]

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

ExtractUncorrelatedTrajectoryData

class propertyestimator.protocols.analysis.**ExtractUncorrelatedTrajectoryData** (*protocol_id*)

A protocol which will subsample frames from a trajectory, yielding only uncorrelated frames as determined from a provided statistical inefficiency and equilibration time.

`__init__` (*protocol_id*)

Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__</code> (<i>protocol_id</i>)	Initialize self.
<code>apply_replicator</code> (<i>replicator</i> , <i>tem-</i> <i>plate_values</i>)	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge</code> (<i>other</i> [, <i>path_replacements</i>])	Determines whether this protocol can be merged with another.
<code>execute</code> (<i>directory</i> , <i>available_resources</i>)	Execute the protocol.
<code>get_class_attribute</code> (<i>reference_path</i>)	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value</code> (<i>reference_path</i>)	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references</code> (<i>input_path</i>)	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.

Continued on next page

Table 140 – continued from previous page

<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>equilibration_index</code>	Protocol Input - The index in the data set after which the data is stationary.
<code>id</code>	The unique id of this protocol.
<code>input_coordinate_file</code>	Protocol Input - The file path to the starting coordinates of a trajectory.
<code>input_trajectory_path</code>	Protocol Input - The file path to the trajectory to subsample.
<code>number_of_uncorrelated_samples</code>	Protocol Output - The number of uncorrelated samples.
<code>output_trajectory_path</code>	Protocol Output - The file path to the subsampled trajectory.
<code>schema</code>	A serializable schema for this object.
<code>statistical_inefficiency</code>	Protocol Input - The statistical inefficiency in the data set.

`input_coordinate_file`

Protocol Input - The file path to the starting coordinates of a trajectory. The default value of this attribute is not set and must be set by the user..

Type str

`input_trajectory_path`

Protocol Input - The file path to the trajectory to subsample. The default value of this attribute is not set and must be set by the user..

Type str

`output_trajectory_path`

Protocol Output - The file path to the subsampled trajectory.

Type str

`execute(directory, available_resources)`

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (`ComputeResources`) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

apply_replicator (*replicator*, *template_values*, *template_index=-1*, *template_value=None*, *update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property_dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

equilibration_index

Protocol Input - The index in the data set after which the data is stationary. The default value of this attribute is not set and must be set by the user..

Type int

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type str

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

number_of_uncorrelated_samples

Protocol Output - The number of uncorrelated samples.

Type int

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path, value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

statistical_inefficiency

Protocol Input - The statistical inefficiency in the data set. The default value of this attribute is not set and must be set by the user..

Type *float*

ExtractUncorrelatedStatisticsData

class `propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData` (*protocol_id*)

A protocol which will subsample entries from a statistics array, yielding only uncorrelated entries as determined from a provided statistical inefficiency and equilibration time.

__init__ (*protocol_id*)

Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__(protocol_id)</code>	Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.

Continued on next page

Table 142 – continued from previous page

<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>equilibration_index</code>	Protocol Input - The index in the data set after which the data is stationary.
<code>id</code>	The unique id of this protocol.
<code>input_statistics_path</code>	Protocol Input - The file path to the statistics to subsample.
<code>number_of_uncorrelated_samples</code>	Protocol Output - The number of uncorrelated samples.
<code>output_statistics_path</code>	Protocol Output - The file path to the subsampled statistics.
<code>schema</code>	A serializable schema for this object.
<code>statistical_inefficiency</code>	Protocol Input - The statistical inefficiency in the data set.

`input_statistics_path`

Protocol Input - The file path to the statistics to subsample. The default value of this attribute is not set and must be set by the user..

Type `str`

`output_statistics_path`

Protocol Output - The file path to the subsampled statistics.

Type `str`

`execute(directory, available_resources)`

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (`str`) – The directory to store output data in.
- **available_resources** (`ComputeResources`) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

apply_replicator (*replicator*, *template_values*, *template_index=-1*, *template_value=None*, *update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property_dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

equilibration_index

Protocol Input - The index in the data set after which the data is stationary. The default value of this attribute is not set and must be set by the user..

Type int

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type str

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type Dict[str, str]

number_of_uncorrelated_samples

Protocol Output - The number of uncorrelated samples.

Type int

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

statistical_inefficiency

Protocol Input - The statistical inefficiency in the data set. The default value of this attribute is not set and must be set by the user..

Type float

Reweighting

<i>ConcatenateTrajectories</i>	A protocol which concatenates multiple trajectories into a single one.
<i>ConcatenateStatistics</i>	A protocol which concatenates multiple trajectories into a single one.
<i>CalculateReducedPotentialOpenMM</i>	Calculates the reduced potential for a given set of configurations.
<i>BaseMBARProtocol</i>	Reweights a set of observables using MBAR to calculate the average value of the observables at a different state than they were originally measured.
<i>ReweightStatistics</i>	Reweights a set of observables from a <i>StatisticsArray</i> using MBAR.

ConcatenateTrajectories

class propertyestimator.protocols.reweighting.**ConcatenateTrajectories** (*protocol_id*)
 A protocol which concatenates multiple trajectories into a single one.

__init__ (*protocol_id*)

Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__(protocol_id)</code>		Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	tem-	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>		Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>		Execute the protocol.
<code>get_class_attribute(reference_path)</code>		Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>		Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>		Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>		Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>		Finds each input which came from a given protocol
<code>set_uuid(value)</code>		Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>		Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>id</code>	The unique id of this protocol.
<code>input_coordinate_paths</code>	Protocol Input - A list of paths to the starting PDB coordinates for each of the trajectories.
<code>input_trajectory_paths</code>	Protocol Input - A list of paths to the trajectories to concatenate.
<code>output_coordinate_path</code>	Protocol Output - The path the PDB coordinate file which contains the topology of the concatenated trajectory.
<code>output_trajectory_path</code>	Protocol Output - The path to the concatenated trajectory.
<code>schema</code>	A serializable schema for this object.

`input_coordinate_paths`

Protocol Input - A list of paths to the starting PDB coordinates for each of the trajectories. The default value of this attribute is not set and must be set by the user..

Type list

`input_trajectory_paths`

Protocol Input - A list of paths to the trajectories to concatenate. The default value of this attribute is not set and must be set by the user..

Type list

output_coordinate_path

Protocol Output - The path the PDB coordinate file which contains the topology of the concatenated trajectory.

Type `str`

output_trajectory_path

Protocol Output - The path to the concatenated trajectory.

Type `str`

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (`ComputeResources`) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a `ProtocolReplicator` to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format `$(replicator.id)`).

Parameters

- **replicator** (`ProtocolReplicator`) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with `template_index` and `template_value`
- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with `template_values` and must be set along with a `template_value`.
- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with `template_values` and must be set along with a `template_index`.
- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other*, *path_replacements*=None)

Determines whether this protocol can be merged with another.

Parameters

- **other** (BaseProtocol) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (ProtocolPath) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (ProtocolPath) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a *list / dict* which contains at least one ProtocolPath.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id` (*str*) – The id of the old input protocol.
- `new_id` (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters `value` (*str*) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- `reference_path` (`ProtocolPath`) – The path pointing to the value to return.
- `value` (*Any*) – The value to set.

ConcatenateStatistics

class `propertyestimator.protocols.reweighting.ConcatenateStatistics` (*protocol_id*)

A protocol which concatenates multiple trajectories into a single one.

`__init__` (*protocol_id*)

Initialize self. See `help(type(self))` for accurate signature.

Methods

`__init__` (*protocol_id*)

Initialize self.

Continued on next page

Table 147 – continued from previous page

<code>apply_replicator(replicator, plate_values)</code>	tem-	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>		Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>		Execute the protocol.
<code>get_class_attribute(reference_path)</code>		Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>		Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>		Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>		Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>		Finds each input which came from a given protocol
<code>set_uuid(value)</code>		Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>		Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>id</code>	The unique id of this protocol.
<code>input_statistics_paths</code>	Protocol Input - A list of paths to statistics arrays to concatenate.
<code>output_statistics_path</code>	Protocol Output - The path the csv file which contains the concatenated statistics.
<code>schema</code>	A serializable schema for this object.

`input_statistics_paths`

Protocol Input - A list of paths to statistics arrays to concatenate. The default value of this attribute is not set and must be set by the user..

Type list

`output_statistics_path`

Protocol Output - The path the csv file which contains the concatenated statistics.

Type str

`execute(directory, available_resources)`

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

apply_replicator (*replicator*, *template_values*, *template_index=-1*, *template_value=None*, *update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property_dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (ProtocolPath) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (ProtocolPath) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a *list / dict* which contains at least one ProtocolPath.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type str

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (BaseProtocol) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type Dict[str, str]

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path, value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

CalculateReducedPotentialOpenMM

class propertyestimator.protocols.reweighting.**CalculateReducedPotentialOpenMM** (*protocol_id*)

Calculates the reduced potential for a given set of configurations.

__init__ (*protocol_id*)

Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__(protocol_id)</code>		Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	tem-	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>		Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>		Execute the protocol.
<code>get_class_attribute(reference_path)</code>		Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>		Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>		Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>		Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>		Finds each input which came from a given protocol
<code>set_uuid(value)</code>		Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>		Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>coordinate_file_path</code>	Protocol Input - The path to the coordinate file which contains topology information about the system.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>enable_pbc</code>	Protocol Input - If true, periodic boundary conditions will be enabled.
<code>high_precision</code>	Protocol Input - If true, OpenMM will be run in double precision mode.
<code>id</code>	The unique id of this protocol.
<code>kinetic_energies_path</code>	Protocol Input - The file path to a statistics array which contain the kinetic energies of each frame in the trajectory.
<code>schema</code>	A serializable schema for this object.
<code>statistics_file_path</code>	Protocol Output - A file path to the statistics file which contains the reduced potentials, and the potential, kinetic and total energies and enthalpies evaluated at the specified state and using the specified system object.
<code>system_path</code>	Protocol Input - The path to the system object which describes the systems potential energy function.
<code>thermodynamic_state</code>	Protocol Input - The state to calculate the reduced potential at.
<code>trajectory_file_path</code>	Protocol Input - The path to the trajectory file which contains the configurations to calculate the energies of.
<code>use_internal_energy</code>	Protocol Input - If true the internal energy, rather than the potential energy will be used when calculating the reduced potential.

thermodynamic_state

Protocol Input - The state to calculate the reduced potential at. The default value of this attribute is not set and must be set by the user..

Type *ThermodynamicState*

system_path

Protocol Input - The path to the system object which describes the systems potential energy function. The default value of this attribute is not set and must be set by the user..

Type str

enable_pbc

Protocol Input - If true, periodic boundary conditions will be enabled. The default value of this attribute is True.

Type bool

coordinate_file_path

Protocol Input - The path to the coordinate file which contains topology information about the system. The default value of this attribute is not set and must be set by the user..

Type str

trajectory_file_path

Protocol Input - The path to the trajectory file which contains the configurations to calculate the energies of. The default value of this attribute is not set and must be set by the user..

Type `str`

kinetic_energies_path

Protocol Input - The file path to a statistics array which contain the kinetic energies of each frame in the trajectory. The default value of this attribute is not set and must be set by the user..

Type `str`

high_precision

Protocol Input - If true, OpenMM will be run in double precision mode. The default value of this attribute is `False`.

Type `bool`

use_internal_energy

Protocol Input - If true the internal energy, rather than the potential energy will be used when calculating the reduced potential. This is required when reweighting properties which depend on the total energy, such as enthalpy. The default value of this attribute is `False`.

Type `bool`

statistics_file_path

Protocol Output - A file path to the statistics file which contains the reduced potentials, and the potential, kinetic and total energies and enthalpies evaluated at the specified state and using the specified system object.

Type `str`

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int*, *optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any*, *optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str*, *optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type *bool*

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type *object*

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input / output

Return type *Any*

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters `input_path` (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id` (*str*) – The id of the old input protocol.
- `new_id` (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters `value` (*str*) – The uuid of the parent calculation.

set_value (*reference_path, value*)

Sets the value of one of this protocols inputs.

Parameters

- `reference_path` (`ProtocolPath`) – The path pointing to the value to return.

- **value** (*Any*) – The value to set.

BaseMBARProtocol

class `propertyestimator.protocols.reweighting.BaseMBARProtocol` (*protocol_id*)
 Reweights a set of observables using MBAR to calculate the average value of the observables at a different state than they were originally measured.

__init__ (*protocol_id*)
 Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__(protocol_id)</code>		Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	tem-	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>		Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>		Execute the protocol.
<code>get_class_attribute(reference_path)</code>		Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>		Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>		Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>		Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>		Finds each input which came from a given protocol
<code>set_uuid(value)</code>		Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>		Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>bootstrap_iterations</code>	Protocol Input - The number of bootstrap iterations to perform if bootstrapped uncertainties have been requested The default value of this attribute is 1.
<code>bootstrap_sample_size</code>	Protocol Input - The relative bootstrap sample size to use if bootstrapped uncertainties have been requested The default value of this attribute is 1.0.
<code>bootstrap_uncertainties</code>	Protocol Input - If true, bootstrapping will be used to estimated the total uncertainty The default value of this attribute is <code>False</code> .
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>effective_sample_indices</code>	Protocol Output - The indices of those samples which have a non-zero weight.

Continued on next page

Table 152 – continued from previous page

<i>effective_samples</i>	Protocol Output - The number of effective samples which were reweighted.
<i>id</i>	The unique id of this protocol.
<i>reference_reduced_potentials</i>	Protocol Input - A list of paths to the reduced potentials of each reference state.
<i>required_effective_samples</i>	Protocol Input - The minimum number of MBAR effective samples for the reweighted value to be trusted.
<i>schema</i>	A serializable schema for this object.
<i>target_reduced_potentials</i>	Protocol Input - A list of paths to the reduced potentials of the target state.
<i>value</i>	Protocol Output - The reweighted average value of the observable at the target state.

reference_reduced_potentials

Protocol Input - A list of paths to the reduced potentials of each reference state. The default value of this attribute is not set and must be set by the user.

Type `list`

target_reduced_potentials

Protocol Input - A list of paths to the reduced potentials of the target state. The default value of this attribute is not set and must be set by the user.

Type `list`

bootstrap_uncertainties

Protocol Input - If true, bootstrapping will be used to estimated the total uncertainty The default value of this attribute is `False`.

Type `bool`

bootstrap_iterations

Protocol Input - The number of bootstrap iterations to perform if bootstrapped uncertainties have been requested The default value of this attribute is `1`.

Type `int`

bootstrap_sample_size

Protocol Input - The relative bootstrap sample size to use if bootstrapped uncertainties have been requested The default value of this attribute is `1.0`.

Type `float`

required_effective_samples

Protocol Input - The minimum number of MBAR effective samples for the reweighted value to be trusted. If this minimum is not met then the uncertainty will be set to `sys.float_info.max` The default value of this attribute is `50`.

Type `int`

value

Protocol Output - The reweighted average value of the observable at the target state.

Type `EstimatedQuantity`

effective_samples

Protocol Output - The number of effective samples which were reweighted.

Type `float`

effective_sample_indices

Protocol Output - The indices of those samples which have a non-zero weight.

Type `list`

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (`ComputeResources`) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a `ProtocolReplicator` to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format $\$(replicator.id)$).

Parameters

- **replicator** (`ProtocolReplicator`) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (`BaseProtocol`) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type `bool`

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (`old_id`, `new_id`)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id` (`str`) – The id of the old input protocol.
- `new_id` (`str`) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (`value`)

Store the uuid of the calculation this protocol belongs to

Parameters `value` (`str`) – The uuid of the parent calculation.

set_value (`reference_path`, `value`)

Sets the value of one of this protocols inputs.

Parameters

- `reference_path` (`ProtocolPath`) – The path pointing to the value to return.
- `value` (`Any`) – The value to set.

ReweightStatistics

class `propertyestimator.protocols.reweighting.ReweightStatistics` (`protocol_id`)

Reweights a set of observables from a `StatisticsArray` using MBAR.

__init__ (`protocol_id`)

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__</code> (<code>protocol_id</code>)	Initialize self.
<code>apply_replicator</code> (<code>replicator</code> , <code>plate_values</code>)	Applies a <code>ProtocolReplicator</code> to this protocol.
<code>can_merge</code> (<code>other</code> [, <code>path_replacements</code>])	Determines whether this protocol can be merged with another.

Continued on next page

Table 153 – continued from previous page

<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>bootstrap_iterations</code>	Protocol Input - The number of bootstrap iterations to perform if bootstrapped uncertainties have been requested The default value of this attribute is 1.
<code>bootstrap_sample_size</code>	Protocol Input - The relative bootstrap sample size to use if bootstrapped uncertainties have been requested The default value of this attribute is 1.0.
<code>bootstrap_uncertainties</code>	Protocol Input - If true, bootstrapping will be used to estimated the total uncertainty The default value of this attribute is <code>False</code> .
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>effective_sample_indices</code>	Protocol Output - The indices of those samples which have a non-zero weight.
<code>effective_samples</code>	Protocol Output - The number of effective samples which were reweighted.
<code>frame_counts</code>	Protocol Input - A list which describes how many of the statistics in the array belong to each reference state.
<code>id</code>	The unique id of this protocol.
<code>reference_reduced_potentials</code>	Protocol Input - A list of paths to the reduced potentials of each reference state.
<code>required_effective_samples</code>	Protocol Input - The minimum number of MBAR effective samples for the reweighted value to be trusted.
<code>schema</code>	A serializable schema for this object.
<code>statistics_paths</code>	Protocol Input - The file paths to the statistics array which contains the observables of interest from each state.
<code>statistics_type</code>	Protocol Input - The type of observable to reweight.
<code>target_reduced_potentials</code>	Protocol Input - A list of paths to the reduced potentials of the target state.

Continued on next page

Table 154 – continued from previous page

<i>value</i>	Protocol Output - The reweighted average value of the observable at the target state.
--------------	--

statistics_paths

Protocol Input - The file paths to the statistics array which contains the observables of interest from each state. If the observable of interest is dependant on the changing variable (e.g. the potential energy) then this must be a path to the observable re-evaluated at the new state. The default value of this attribute is not set and must be set by the user..

Type *list*

statistics_type

Protocol Input - The type of observable to reweight. The default value of this attribute is not set and must be set by the user..

Type *ObservableType*

frame_counts

Protocol Input - A list which describes how many of the statistics in the array belong to each reference state. If this input is used, only a single file path should be passed to the *statistics_paths* input. The default value of this attribute is []. This input is *optional*.

Type *list*

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type *Dict[str, Any]*

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is *True*.

Type *bool*

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

bootstrap_iterations

Protocol Input - The number of bootstrap iterations to perform if bootstrapped uncertainties have been requested The default value of this attribute is 1.

Type int

bootstrap_sample_size

Protocol Input - The relative bootstrap sample size to use if bootstrapped uncertainties have been requested The default value of this attribute is 1 . 0.

Type float

bootstrap_uncertainties

Protocol Input - If true, bootstrapping will be used to estimated the total uncertainty The default value of this attribute is False.

Type bool

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property_dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

effective_sample_indices

Protocol Output - The indices of those samples which have a non-zero weight.

Type list

effective_samples

Protocol Output - The number of effective samples which were reweighted.

Type float

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type str

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

reference_reduced_potentials

Protocol Input - A list of paths to the reduced potentials of each reference state. The default value of this attribute is not set and must be set by the user.

Type list

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

required_effective_samples

Protocol Input - The minimum number of MBAR effective samples for the reweighted value to be trusted. If this minimum is not met then the uncertainty will be set to `sys.float_info.max`. The default value of this attribute is 50.

Type `int`

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (value)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (reference_path, value)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (`ProtocolPath`) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

target_reduced_potentials

Protocol Input - A list of paths to the reduced potentials of the target state. The default value of this attribute is not set and must be set by the user..

Type `list`

value

Protocol Output - The reweighted average value of the observable at the target state.

Type `EstimatedQuantity`

Gradients

<i>GradientReducedPotentials</i>	A protocol to estimates the the reduced potential of the configurations of a trajectory using reverse and forward perturbed simulation parameters for use with estimating reweighted gradients using the central difference method.
<i>CentralDifferenceGradient</i>	A protocol which employs the central difference method to estimate the gradient of an observable A, such that

GradientReducedPotentials

class `propertyestimator.protocols.gradients.GradientReducedPotentials` (*protocol_id*)
 A protocol to estimates the the reduced potential of the configurations of a trajectory using reverse and forward perturbed simulation parameters for use with estimating reweighted gradients using the central difference method.

`__init__` (*protocol_id*)
 Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__(protocol_id)</code>	Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>coordinate_file_path</code>	Protocol Input - A path to a PDB coordinate file which describes the topology of the system.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>effective_sample_indices</code>	Protocol Input - This a placeholder input which is not currently implemented.
<code>enable_pbc</code>	Protocol Input - If true, periodic boundary conditions will be enabled when re-evaluating the reduced potentials.
<code>force_field_path</code>	Protocol Input - The path to the force field which contains the parameters to differentiate the observable with respect to.
<code>forward_parameter_value</code>	Protocol Output - The value of the parameter perturbed in the forward direction.

Continued on next page

Table 157 – continued from previous page

<i>forward_potentials_path</i>	Protocol Output - A file path to the energies evaluated using the parameters perturbed in the forward direction.
<i>id</i>	The unique id of this protocol.
<i>parameter_key</i>	Protocol Input - The key of the parameter to differentiate with respect to.
<i>perturbation_scale</i>	Protocol Input - The amount to perturb the parameter by, such that $p_{\text{new}} = p_{\text{old}} * (1 \pm \text{perturbation_scale})$ The default value of this attribute is 0.0001.
<i>reference_force_field_paths</i>	Protocol Input - A list of paths to the force field files which were originally used to generate the configurations.
<i>reference_potential_paths</i>	Protocol Output - File paths to the reduced potentials evaluated using each of the reference force fields.
<i>reference_statistics_path</i>	Protocol Input - An optional path to the statistics array which was generated alongside the observable of interest, which will be used to correct the potential energies at the reverse and forward states.
<i>reverse_parameter_value</i>	Protocol Output - The value of the parameter perturbed in the reverse direction.
<i>reverse_potentials_path</i>	Protocol Output - A file path to the energies evaluated using the parameters perturbed in the reverse direction.
<i>schema</i>	A serializable schema for this object.
<i>substance</i>	Protocol Input - The substance which describes the composition of the system.
<i>thermodynamic_state</i>	Protocol Input - The thermodynamic state to estimate the gradients at.
<i>trajectory_file_path</i>	Protocol Input - A path to the trajectory of configurations The default value of this attribute is not set and must be set by the user..
<i>use_subset_of_force_field</i>	Protocol Input - If true, the reduced potential will be estimated using an OpenMM system which only contains the parameter of interest The default value of this attribute is True.

reference_force_field_paths

Protocol Input - A list of paths to the force field files which were originally used to generate the configurations. The default value of this attribute is not set and must be set by the user..

Type `list`

force_field_path

Protocol Input - The path to the force field which contains the parameters to differentiate the observable with respect to. The default value of this attribute is not set and must be set by the user..

Type `str`

reference_statistics_path

Protocol Input - An optional path to the statistics array which was generated alongside the observable of interest, which will be used to correct the potential energies at the reverse and forward states. This is only really needed when the observable of interest is an energy. The default value of this attribute is not set.

This input is *optional*.

Type `str`

enable_pbc

Protocol Input - If true, periodic boundary conditions will be enabled when re-evaluating the reduced potentials. The default value of this attribute is `True`.

Type `bool`

substance

Protocol Input - The substance which describes the composition of the system. The default value of this attribute is not set and must be set by the user.

Type `Substance`

thermodynamic_state

Protocol Input - The thermodynamic state to estimate the gradients at. The default value of this attribute is not set and must be set by the user.

Type `ThermodynamicState`

coordinate_file_path

Protocol Input - A path to a PDB coordinate file which describes the topology of the system. The default value of this attribute is not set and must be set by the user.

Type `str`

trajectory_file_path

Protocol Input - A path to the trajectory of configurations The default value of this attribute is not set and must be set by the user.

Type `str`

parameter_key

Protocol Input - The key of the parameter to differentiate with respect to. The default value of this attribute is not set and must be set by the user.

Type `ParameterGradientKey`

perturbation_scale

Protocol Input - The amount to perturb the parameter by, such that $p_{\text{new}} = p_{\text{old}} * (1 \pm \text{perturbation_scale})$ The default value of this attribute is `0.0001`.

Type `float`

use_subset_of_force_field

Protocol Input - If true, the reduced potential will be estimated using an OpenMM system which only contains the parameter of interest The default value of this attribute is `True`.

Type `bool`

effective_sample_indices

Protocol Input - This a placeholder input which is not currently implemented. The default value of this attribute is not set. This input is *optional*.

Type `list`

reference_potential_paths

Protocol Output - File paths to the reduced potentials evaluated using each of the reference force fields.

Type `list`

reverse_potentials_path

Protocol Output - A file path to the energies evaluated using the parameters perturbed in the reverse direction.

Type `str`

forward_potentials_path

Protocol Output - A file path to the energies evaluated using the parameters perturbed in the forward direction.

Type `str`

reverse_parameter_value

Protocol Output - The value of the parameter perturbed in the reverse direction.

Type `Quantity`

forward_parameter_value

Protocol Output - The value of the parameter perturbed in the forward direction.

Type `Quantity`

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (`ComputeResources`) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a `ProtocolReplicator` to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format `$(replicator.id)`).

Parameters

- **replicator** (`ProtocolReplicator`) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with `template_index` and `template_value`

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with `template_values` and must be set along with a `template_value`.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters `input_path` (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

`property id`

The unique id of this protocol.

Type `str`

`merge (other)`

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (*BaseProtocol*) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

`replace_protocol (old_id, new_id)`

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id` (*str*) – The id of the old input protocol.
- `new_id` (*str*) – The id of the new input protocol.

`property schema`

A serializable schema for this object.

Type *ProtocolSchema*

`set_uuid (value)`

Store the uuid of the calculation this protocol belongs to

Parameters `value` (*str*) – The uuid of the parent calculation.

`set_value (reference_path, value)`

Sets the value of one of this protocols inputs.

Parameters

- `reference_path` (*ProtocolPath*) – The path pointing to the value to return.
- `value` (*Any*) – The value to set.

CentralDifferenceGradient

class `propertyestimator.protocols.gradients.CentralDifferenceGradient` (*protocol_id*)

A protocol which employs the central difference method to estimate the gradient of an observable A, such that

$$\text{grad} = (A(x-h) - A(x+h)) / (2h)$$

Notes

The *values* input must either be a list of `unit.Quantity`, a `ProtocolPath` to a list of `unit.Quantity`, or a list of `ProtocolPath` which each point to a `unit.Quantity`.

`__init__` (*protocol_id*)

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__(protocol_id)</code>	Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another <i>BaseProtocol</i> with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>forward_observable_value</code>	Protocol Input - The value of the observable evaluated using the parameters perturbed in the forward direction.
<code>forward_parameter_value</code>	Protocol Input - The value of the parameter perturbed in the forward direction.
<code>gradient</code>	Protocol Output - The estimated gradient
<code>id</code>	The unique id of this protocol.

Continued on next page

Table 159 – continued from previous page

<i>parameter_key</i>	Protocol Input - The key of the parameter to differentiate with respect to.
<i>reverse_observable_value</i>	Protocol Input - The value of the observable evaluated using the parameters perturbed in the reverse direction.
<i>reverse_parameter_value</i>	Protocol Input - The value of the parameter perturbed in the reverse direction.
<i>schema</i>	A serializable schema for this object.

parameter_key

Protocol Input - The key of the parameter to differentiate with respect to. The default value of this attribute is not set and must be set by the user..

Type *ParameterGradientKey*

reverse_observable_value

Protocol Input - The value of the observable evaluated using the parameters perturbed in the reverse direction. The default value of this attribute is not set and must be set by the user..

Type typing.Union[pint.quantity.build_quantity_class.<locals>.Quantity, propertyestimator.utils.quantities.EstimatedQuantity]

forward_observable_value

Protocol Input - The value of the observable evaluated using the parameters perturbed in the forward direction. The default value of this attribute is not set and must be set by the user..

Type typing.Union[pint.quantity.build_quantity_class.<locals>.Quantity, propertyestimator.utils.quantities.EstimatedQuantity]

reverse_parameter_value

Protocol Input - The value of the parameter perturbed in the reverse direction. The default value of this attribute is not set and must be set by the user..

Type Quantity

forward_parameter_value

Protocol Input - The value of the parameter perturbed in the forward direction. The default value of this attribute is not set and must be set by the user..

Type Quantity

gradient

Protocol Output - The estimated gradient

Type *ParameterGradient*

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator*, *template_values*, *template_index=-1*, *template_value=None*, *update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type `bool`

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type `Any`

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type `dict` of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.

- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path, value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

Groups

<i>ProtocolGroup</i>	A collection of protocols to be executed in one batch.
<i>ConditionalGroup</i>	A collection of protocols which are to execute until a given condition is met.

ProtocolGroup

class `propertyestimator.protocols.groups.ProtocolGroup` (*protocol_id*)

A collection of protocols to be executed in one batch.

This may be used for example to cluster together multiple protocols that will execute in a linear chain so that multiple scheduler execution calls are reduced into a single one.

Additionally, a group may provide enhanced behaviour, for example running all protocols within the group self consistently until a given condition is met (e.g run a simulation until a given observable has converged).

__init__ (*protocol_id*)

Constructs a new ProtocolGroup.

Methods

__init__ (<i>protocol_id</i>)	Constructs a new ProtocolGroup.
add_protocols (* <i>protocols</i>)	
apply_replicator (<i>replicator, plate_values</i>)	Applies a <i>ProtocolReplicator</i> to this protocol.
can_merge (<i>other[, path_replacements]</i>)	Determines whether this protocol group can be merged with another.
execute (<i>directory, available_resources</i>)	Executes the protocols within this groups
get_class_attribute (<i>reference_path</i>)	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
get_value (<i>reference_path</i>)	Returns the value of one of this protocols parameters / inputs.

Continued on next page

Table 161 – continued from previous page

<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another ProtocolGroup with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols parameters / inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependants_graph</code>	A dictionary of which stores which grouped protocols are dependant on other grouped protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>execution_order</code>	The ids of the protocols in the group, in the order in which they will be internally executed.
<code>id</code>	The unique id of this protocol.
<code>protocols</code>	A dictionary of the protocols in this groups, where the dictionary key is the protocol id, and the value the protocol itself.
<code>root_protocols</code>	The ids of the protocols in the group which do not take input from the other grouped protocols.
<code>schema</code>	A serializable schema for this object.

property root_protocols

The ids of the protocols in the group which do not take input from the other grouped protocols.

Type List[str]

property execution_order

The ids of the protocols in the group, in the order in which they will be internally executed.

Type List[str]

property dependants_graph

A dictionary of which stores which grouped protocols are dependant on other grouped protocols. Each key in the dictionary is the id of a grouped protocol, and each value is the id of a protocol which depends on the protocol by the key.

Type Dict[str, str]

property protocols

A dictionary of the protocols in this groups, where the dictionary key is the protocol id, and the value the protocol itself.

Type Dict[str, *BaseProtocol*]

set_uuid (value)

Store the uuid of the calculation this protocol belongs to

Parameters *value* (str) – The uuid of the parent calculation.

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a different one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

execute (*directory, available_resources*)

Executes the protocols within this groups

Parameters

- **directory** (*str*) – The root directory in which to run the protocols
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns True if all the protocols execute correctly.

Return type `bool`

can_merge (*other, path_replacements=None*)

Determines whether this protocol group can be merged with another.

Parameters

- **other** (*ProtocolGroup*) – The protocol group to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type `bool`

merge (*other*)

Merges another ProtocolGroup with this one. The id of this protocol will remain unchanged.

It is assumed that can_merge has already returned that these protocol groups are compatible to be merged together.

Parameters **other** (*ProtocolGroup*) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols parameters / inputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input

Return type object

set_value (*reference_path*, *value*)

Sets the value of one of this protocols parameters / inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

apply_replicator (*replicator*, *template_values*, *template_index=-1*, *template_value=None*, *update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

property_dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a list / dict which contains at least one ProtocolPath.

Parameters *input_path* (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type str

property schema

A serializable schema for this object.

Type ProtocolSchema

ConditionalGroup

class propertyestimator.protocols.groups.**ConditionalGroup** (*protocol_id*)

A collection of protocols which are to execute until a given condition is met.

__init__ (*protocol_id*)

Constructs a new ConditionalGroup

Methods

<code>__init__(protocol_id)</code>	Constructs a new ConditionalGroup
<code>add_condition(condition_to_add)</code>	Adds a condition to this groups list of conditions if it not already in the condition list.
<code>add_protocols(*protocols)</code>	
<code>apply_replicator(replicator, plate_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol group can be merged with another.
<code>execute(directory, available_resources)</code>	Executes the protocols within this groups
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols parameters / inputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another ProtocolGroup with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol

Continued on next page

Table 163 – continued from previous page

<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols parameters / inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>conditions</code>	
<code>current_iteration</code>	Protocol Output - The current number of iterations this group has performed while attempting to satisfy the specified conditions.
<code>dependants_graph</code>	A dictionary of which stores which grouped protocols are dependant on other grouped protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>execution_order</code>	The ids of the protocols in the group, in the order in which they will be internally executed.
<code>id</code>	The unique id of this protocol.
<code>max_iterations</code>	Protocol Input - The maximum number of iterations to run for to try and satisfy the groups conditions.
<code>protocols</code>	A dictionary of the protocols in this groups, where the dictionary key is the protocol id, and the value the protocol itself.
<code>root_protocols</code>	The ids of the protocols in the group which do not take input from the other grouped protocols.
<code>schema</code>	A serializable schema for this object.

class ConditionType

The acceptable conditions to place on the group

max_iterations

Protocol Input - The maximum number of iterations to run for to try and satisfy the groups conditions. The default value of this attribute is 100.

Type int

current_iteration

Protocol Output - The current number of iterations this group has performed while attempting to satisfy the specified conditions. This value starts from one.

Type int

execute (*directory, available_resources*)

Executes the protocols within this groups

Parameters

- **directory** (*str*) – The root directory in which to run the protocols
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns True if all the protocols execute correctly.

Return type `bool`

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol group can be merged with another.

Parameters

- **other** (`ProtocolGroup`) – The protocol group to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type `bool`

merge (*other*)

Merges another ProtocolGroup with this one. The id of this protocol will remain unchanged.

It is assumed that can_merge has already returned that these protocol groups are compatible to be merged together.

Parameters **other** (`ConditionalGroup`) – The protocol to merge into this one.

add_condition (*condition_to_add*)

Adds a condition to this groups list of conditions if it not already in the condition list.

Parameters **condition_to_add** (`ConditionalGroup.Condition`) – The condition to add.

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a different one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols parameters / inputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input

Return type `object`

set_value (*reference_path, value*)

Sets the value of one of this protocols parameters / inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a *ProtocolPath*, or a *list / dict* which contains at least one *ProtocolPath*.

Parameters *input_path* (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of *ProtocolPath* and *ProtocolPath*

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is *True*.

Type *bool*

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

property dependants_graph

A dictionary of which stores which grouped protocols are dependant on other grouped protocols. Each key in the dictionary is the id of a grouped protocol, and each value is the id of a protocol which depends on the protocol by the key.

Type Dict[str, str]

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

property execution_order

The ids of the protocols in the group, in the order in which they will be internally executed.

Type List[str]

property id

The unique id of this protocol.

Type str

property protocols

A dictionary of the protocols in this groups, where the dictionary key is the protocol id, and the value the protocol itself.

Type Dict[str, *BaseProtocol*]

property root_protocols

The ids of the protocols in the group which do not take input from the other grouped protocols.

Type List[str]

property schema

A serializable schema for this object.

Type *ProtocolSchema*

Storage

<i>UnpackStoredDataCollection</i>	Loads a <i>StoredDataCollection</i> object from disk, and makes its inner data objects easily accessible to other protocols.
<i>UnpackStoredSimulationData</i>	Loads a <i>StoredSimulationData</i> object from disk, and makes its attributes easily accessible to other protocols.

UnpackStoredDataCollection

class propertyestimator.protocols.storage.**UnpackStoredDataCollection** (*protocol_id*)
 Loads a *StoredDataCollection* object from disk, and makes its inner data objects easily accessible to other protocols.

__init__ (*protocol_id*)

Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__(protocol_id)</code>		Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	tem-	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>		Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>		Execute the protocol.
<code>get_class_attribute(reference_path)</code>		Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>		Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>		Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>		Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>		Finds each input which came from a given protocol
<code>set_uuid(value)</code>		Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>		Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>collection_data_paths</code>	Protocol Output - A dictionary of data object path, data directory path and force field path tuples partitioned by the unique collection keys.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>id</code>	The unique id of this protocol.
<code>input_data_path</code>	Protocol Input - A tuple which contains both the path to the simulation data object, it's ancillary data directory, and the force field which was used to generate the stored data.
<code>schema</code>	A serializable schema for this object.

input_data_path

Protocol Input - A tuple which contains both the path to the simulation data object, it's ancillary data directory, and the force field which was used to generate the stored data. The default value of this attribute is not set and must be set by the user..

Type typing.Union[list, tuple]

collection_data_paths

Protocol Output - A dictionary of data object path, data directory path and force field path tuples partitioned by the unique collection keys.

Type dict

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (`BaseProtocol`) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type `bool`

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of `ProtocolPath`

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children’s, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type Dict[str, str]

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

UnpackStoredSimulationData

class `propertyestimator.protocols.storage.UnpackStoredSimulationData` (*protocol_id*)
Loads a *StoredSimulationData* object from disk, and makes its attributes easily accessible to other protocols.

__init__ (*protocol_id*)

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__(protocol_id)</code>		Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	tem-	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>		Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>		Execute the protocol.
<code>get_class_attribute(reference_path)</code>		Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>		Returns the value of one of this protocols inputs / outputs.

Continued on next page

Table 168 – continued from previous page

<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>coordinate_file_path</code>	Protocol Output - A path to the stored simulation output coordinates.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>force_field_path</code>	Protocol Output - A path to the force field parameters used to generate the stored data.
<code>id</code>	The unique id of this protocol.
<code>schema</code>	A serializable schema for this object.
<code>simulation_data_path</code>	Protocol Input - A list / tuple which contains both the path to the simulation data object, it's ancillary data directory, and the force field which was used to generate the stored data.
<code>statistical_inefficiency</code>	Protocol Output - The statistical inefficiency of the stored data.
<code>statistics_file_path</code>	Protocol Output - A path to the stored simulation statistics array.
<code>substance</code>	Protocol Output - The substance which was stored.
<code>thermodynamic_state</code>	Protocol Output - The thermodynamic state which was stored.
<code>total_number_of_molecules</code>	Protocol Output - The total number of molecules in the stored system.
<code>trajectory_file_path</code>	Protocol Output - A path to the stored simulation trajectory.

`simulation_data_path`

Protocol Input - A list / tuple which contains both the path to the simulation data object, it's ancillary data directory, and the force field which was used to generate the stored data. The default value of this attribute is not set and must be set by the user..

Type `typing.Union[list, tuple]`

`substance`

Protocol Output - The substance which was stored.

Type `Substance`

`total_number_of_molecules`

Protocol Output - The total number of molecules in the stored system.

Type `int`

thermodynamic_state

Protocol Output - The thermodynamic state which was stored.

Type *ThermodynamicState*

statistical_inefficiency

Protocol Output - The statistical inefficiency of the stored data.

Type *float*

coordinate_file_path

Protocol Output - A path to the stored simulation output coordinates.

Type *str*

trajectory_file_path

Protocol Output - A path to the stored simulation trajectory.

Type *str*

statistics_file_path

Protocol Output - A path to the stored simulation statistics array.

Type *str*

force_field_path

Protocol Output - A path to the force field parameters used to generate the stored data.

Type *str*

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[*str*, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is *True*.

Type *bool*

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters `input_path` (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id` (*str*) – The id of the old input protocol.
- `new_id` (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters `value` (*str*) – The uuid of the parent calculation.

set_value (*reference_path, value*)

Sets the value of one of this protocols inputs.

Parameters

- `reference_path` (`ProtocolPath`) – The path pointing to the value to return.

- **value** (*Any*) – The value to set.

Miscellaneous

<i>AddValues</i>	A protocol to add together a list of values.
<i>SubtractValues</i>	A protocol to subtract one value from another such that:
<i>MultiplyValue</i>	A protocol which multiplies a value by a specified scalar
<i>DivideValue</i>	A protocol which divides a value by a specified scalar
<i>FilterSubstanceByRole</i>	A protocol which takes a substance as input, and returns a substance which only contains components whose role match a given criteria.
<i>BaseWeightByMoleFraction</i>	
<i>WeightByMoleFraction</i>	Multiplies a value by the mole fraction of a component in a <i>Substance</i> .

AddValues

class propertyestimator.protocols.miscellaneous.**AddValues** (*protocol_id*)
 A protocol to add together a list of values.

Notes

The *values* input must either be a list of unit.Quantity, a ProtocolPath to a list of unit.Quantity, or a list of ProtocolPath which each point to a unit.Quantity.

`__init__` (*protocol_id*)
 Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__(protocol_id)</code>	Initialize self.
<code>apply_replicator(replicator, tem-plate_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>id</code>	The unique id of this protocol.
<code>result</code>	Protocol Output - The sum of the values.
<code>schema</code>	A serializable schema for this object.
<code>values</code>	Protocol Input - The values to add together.

values

Protocol Input - The values to add together. The default value of this attribute is not set and must be set by the user..

Type list

result

Protocol Output - The sum of the values.

Type typing.Union[int, float, propertyestimator.utils.quantities.EstimatedQuantity, pint.quantity.build_quantity_class.<locals>.Quantity, propertyestimator.properties.properties.ParameterGradient]

execute (directory, available_resources)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

apply_replicator (replicator, template_values, template_index=-1, template_value=None, update_input_references=False)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format $\$(replicator.id)$).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters `input_path` (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id` (*str*) – The id of the old input protocol.
- `new_id` (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters `value` (*str*) – The uuid of the parent calculation.

set_value (*reference_path, value*)

Sets the value of one of this protocols inputs.

Parameters

- `reference_path` (`ProtocolPath`) – The path pointing to the value to return.

- **value** (*Any*) – The value to set.

SubtractValues

class `propertyestimator.protocols.miscellaneous.SubtractValues` (*protocol_id*)

A protocol to subtract one value from another such that:

$result = value_b - value_a$

`__init__` (*protocol_id*)

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__(protocol_id)</code>	Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>id</code>	The unique id of this protocol.
<code>result</code>	Protocol Output - The results of $value_b - value_a$.
<code>schema</code>	A serializable schema for this object.
<code>value_a</code>	Protocol Input - $value_a$ in the formula $result = value_b - value_a$.
<code>value_b</code>	Protocol Input - $value_b$ in the formula $result = value_b - value_a$.

value_a

Protocol Input - $value_a$ in the formula $result = value_b - value_a$. The default value of this attribute is not set and must be set by the user..

Type typing.Union[int, float, pint.quantity.build_quantity_class.<locals>.Quantity, propertyestimator.utils.quantities.EstimatedQuantity, propertyestimator.properties.properties.ParameterGradient]

value_b

Protocol Input - *value_b* in the formula $result = value_b - value_a$. The default value of this attribute is not set and must be set by the user..

Type typing.Union[int, float, pint.quantity.build_quantity_class.<locals>.Quantity, propertyestimator.utils.quantities.EstimatedQuantity, propertyestimator.properties.properties.ParameterGradient]

result

Protocol Output - The results of *value_b* - *value_a*.

Type typing.Union[int, float, propertyestimator.utils.quantities.EstimatedQuantity, pint.quantity.build_quantity_class.<locals>.Quantity, propertyestimator.properties.properties.ParameterGradient]

execute (directory, available_resources)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[*str*, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

apply_replicator (replicator, template_values, template_index=-1, template_value=None, update_input_references=False)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format $\$(replicator.id)$).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type *bool*

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type *object*

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a *list / dict* which contains at least one ProtocolPath.

Parameters `input_path` (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by `input_path` depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type `str`

merge (`other`)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type Dict[`str`, `str`]

replace_protocol (`old_id`, `new_id`)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id` (`str`) – The id of the old input protocol.
- `new_id` (`str`) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (`value`)

Store the uuid of the calculation this protocol belongs to

Parameters `value` (`str`) – The uuid of the parent calculation.

set_value (`reference_path`, `value`)

Sets the value of one of this protocols inputs.

Parameters

- `reference_path` (`ProtocolPath`) – The path pointing to the value to return.
- `value` (`Any`) – The value to set.

MultiplyValue

class `propertyestimator.protocols.miscellaneous.MultiplyValue` (`protocol_id`)

A protocol which multiplies a value by a specified scalar

`__init__(protocol_id)`
 Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__(protocol_id)</code>	Initialize self.
<code>apply_replicator(replicator, template_values)</code>	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>	Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>id</code>	The unique id of this protocol.
<code>multiplier</code>	Protocol Input - The scalar to multiply by.
<code>result</code>	Protocol Output - The result of the multiplication.
<code>schema</code>	A serializable schema for this object.
<code>value</code>	Protocol Input - The value to multiply.

value

Protocol Input - The value to multiply. The default value of this attribute is not set and must be set by the user..

Type typing.Union[int, float, pint.quantity.build_quantity_class.<locals>.Quantity, propertyestimator.utils.quantities.EstimatedQuantity, propertyestimator.properties.properties.ParameterGradient]

multiplier

Protocol Input - The scalar to multiply by. The default value of this attribute is not set and must be set by the user..

Type typing.Union[int, float, pint.quantity.build_quantity_class.<locals>.Quantity]

result

Protocol Output - The result of the multiplication.

Type typing.Union[int, float, propertyestimator.utils.quantities.EstimatedQuantity, pint.quantity.build_quantity_class.<locals>.Quantity, propertyestimator.properties.properties.ParameterGradient]

execute (*directory*, *available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is True.

Type bool

apply_replicator (*replicator*, *template_values*, *template_index=-1*, *template_value=None*, *update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int*, *optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any*, *optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_index* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type bool

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the attribute to return.

Returns The class attribute.

Return type object

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (*ProtocolPath*) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a list / dict which contains at least one ProtocolPath.

Parameters **input_path** (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type str

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (`old_id`, `new_id`)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- `old_id` (`str`) – The id of the old input protocol.
- `new_id` (`str`) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type `ProtocolSchema`

set_uuid (`value`)

Store the uuid of the calculation this protocol belongs to

Parameters `value` (`str`) – The uuid of the parent calculation.

set_value (`reference_path`, `value`)

Sets the value of one of this protocols inputs.

Parameters

- `reference_path` (`ProtocolPath`) – The path pointing to the value to return.
- `value` (`Any`) – The value to set.

DivideValue

class `propertyestimator.protocols.miscellaneous.DivideValue` (`protocol_id`)

A protocol which divides a value by a specified scalar

`__init__` (`protocol_id`)

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__</code> (<code>protocol_id</code>)	Initialize self.
<code>apply_replicator</code> (<code>replicator</code> , <code>tem-plate_values</code>)	Applies a <code>ProtocolReplicator</code> to this protocol.
<code>can_merge</code> (<code>other</code> [, <code>path_replacements</code>])	Determines whether this protocol can be merged with another.

Continued on next page

Table 177 – continued from previous page

<code>execute(directory, available_resources)</code>	Execute the protocol.
<code>get_class_attribute(reference_path)</code>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>	Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>	Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>	Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>	Finds each input which came from a given protocol
<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>divisor</code>	Protocol Input - The scalar to divide by.
<code>id</code>	The unique id of this protocol.
<code>result</code>	Protocol Output - The result of the division.
<code>schema</code>	A serializable schema for this object.
<code>value</code>	Protocol Input - The value to divide.

value

Protocol Input - The value to divide. The default value of this attribute is not set and must be set by the user..

Type typing.Union[int, float, pint.quantity.build_quantity_class.<locals>.Quantity, propertyestimator.utils.quantities.EstimatedQuantity, propertyestimator.properties.properties.ParameterGradient]

divisor

Protocol Input - The scalar to divide by. The default value of this attribute is not set and must be set by the user..

Type typing.Union[int, float, pint.quantity.build_quantity_class.<locals>.Quantity]

result

Protocol Output - The result of the division.

Type typing.Union[int, float, propertyestimator.utils.quantities.EstimatedQuantity, pint.quantity.build_quantity_class.<locals>.Quantity, propertyestimator.properties.properties.ParameterGradient]

execute (directory, available_resources)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.

- **available_resources** (`ComputeResources`) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator*, *template_values*, *template_index=-1*, *template_value=None*, *update_input_references=False*)

Applies a `ProtocolReplicator` to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format `$(replicator.id)`).

Parameters

- **replicator** (`ProtocolReplicator`) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with `template_index` and `template_value`

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with `template_values` and must be set along with a `template_value`.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with `template_values` and must be set along with a `template_index`.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific `template_index` or `template_value` is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the `template_values` array.

Return type `dict` of `ProtocolPath` and list of tuple of `ProtocolPath` and `int`

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (`BaseProtocol`) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns `True` if the two protocols are safe to merge.

Return type `bool`

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters `reference_path` (ProtocolPath) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters `reference_path` (ProtocolPath) – The path pointing to the value to return.

Returns The value of the input / output

Return type Any

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a ProtocolPath, or a *list / dict* which contains at least one ProtocolPath.

Parameters `input_path` (*propertyestimator.workflow.utils.ProtocolPath*) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type dict of ProtocolPath and ProtocolPath

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another BaseProtocol with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters `other` (BaseProtocol) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type Dict[str, str]

replace_protocol (*old_id, new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.
- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

FilterSubstanceByRole

class `propertyestimator.protocols.miscellaneous.FilterSubstanceByRole` (*protocol_id*)

A protocol which takes a substance as input, and returns a substance which only contains components whose role match a given criteria.

__init__ (*protocol_id*)

Initialize self. See `help(type(self))` for accurate signature.

Methods

<code>__init__(protocol_id)</code>		Initialize self.
<code>apply_replicator(replicator, plate_values)</code>	tem-	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge(other[, path_replacements])</code>		Determines whether this protocol can be merged with another.
<code>execute(directory, available_resources)</code>		Execute the protocol.
<code>get_class_attribute(reference_path)</code>		Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value(reference_path)</code>		Returns the value of one of this protocols inputs / outputs.
<code>get_value_references(input_path)</code>		Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge(other)</code>		Merges another BaseProtocol with this one.
<code>replace_protocol(old_id, new_id)</code>		Finds each input which came from a given protocol

Continued on next page

Table 179 – continued from previous page

<code>set_uuid(value)</code>	Store the uuid of the calculation this protocol belongs to
<code>set_value(reference_path, value)</code>	Sets the value of one of this protocols inputs.

Attributes

<code>allow_merging</code>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<code>component_role</code>	Protocol Input - The role to filter substance components against.
<code>dependencies</code>	A list of pointers to the protocols which this protocol takes input from.
<code>expected_components</code>	Protocol Input - The number of components expected to remain after filtering.
<code>filtered_substance</code>	Protocol Output - The filtered substance.
<code>id</code>	The unique id of this protocol.
<code>input_substance</code>	Protocol Input - The substance to filter.
<code>schema</code>	A serializable schema for this object.

input_substance

Protocol Input - The substance to filter. The default value of this attribute is not set and must be set by the user..

Type *Substance*

component_role

Protocol Input - The role to filter substance components against. The default value of this attribute is not set and must be set by the user..

Type *Substance.ComponentRole*

expected_components

Protocol Input - The number of components expected to remain after filtering. An exception is raised if this number is not matched. The default value of this attribute is not set. This input is *optional*.

Type *int*

filtered_substance

Protocol Output - The filtered substance.

Type *Substance*

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type `Dict[str, Any]`

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator, template_values, template_index=-1, template_value=None, update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other, path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type `bool`

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type `Any`

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type `dict` of `ProtocolPath` and `ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (`str`) – The id of the old input protocol.

- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path*, *value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

WeightByMoleFraction

class propertyestimator.protocols.miscellaneous.**WeightByMoleFraction** (*protocol_id*)

Multiplies a value by the mole fraction of a component in a *Substance*.

`__init__` (*protocol_id*)

Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__</code> (<i>protocol_id</i>)		Initialize self.
<code>apply_replicator</code> (<i>replicator</i> , <i>plate_values</i>)	tem-	Applies a <i>ProtocolReplicator</i> to this protocol.
<code>can_merge</code> (<i>other</i> [, <i>path_replacements</i>])		Determines whether this protocol can be merged with another.
<code>execute</code> (<i>directory</i> , <i>available_resources</i>)		Execute the protocol.
<code>get_class_attribute</code> (<i>reference_path</i>)		Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
<code>get_value</code> (<i>reference_path</i>)		Returns the value of one of this protocols inputs / outputs.
<code>get_value_references</code> (<i>input_path</i>)		Returns a dictionary of references to the protocols which one of this protocols inputs (specified by <i>input_path</i>) takes its value from.
<code>merge</code> (<i>other</i>)		Merges another BaseProtocol with this one.
<code>replace_protocol</code> (<i>old_id</i> , <i>new_id</i>)		Finds each input which came from a given protocol
<code>set_uuid</code> (<i>value</i>)		Store the uuid of the calculation this protocol belongs to
<code>set_value</code> (<i>reference_path</i> , <i>value</i>)		Sets the value of one of this protocols inputs.

Attributes

<i>allow_merging</i>	Protocol Input - Defines whether this protocols is allowed to merge with other protocols.
<i>component</i>	Protocol Input - The component whose mole fraction to weight by.
<i>dependencies</i>	A list of pointers to the protocols which this protocol takes input from.
<i>full_substance</i>	Protocol Input - The full substance which describes the mole fraction of the component.
<i>id</i>	The unique id of this protocol.
<i>schema</i>	A serializable schema for this object.
<i>value</i>	Protocol Input - The value to be weighted.
<i>weighted_value</i>	Protocol Output - The value weighted by the <i>component's mole fraction as determined from the 'full_substance</i> .

value

Protocol Input - The value to be weighted. The default value of this attribute is not set and must be set by the user..

Type typing.Union[float, int, propertyestimator.utils.quantities.EstimatedQuantity, pint.quantity.build_quantity_class.<locals>.Quantity, propertyestimator.properties.properties.ParameterGradient]

component

Protocol Input - The component whose mole fraction to weight by. The default value of this attribute is not set and must be set by the user..

Type *Substance*

full_substance

Protocol Input - The full substance which describes the mole fraction of the component. The default value of this attribute is not set and must be set by the user..

Type *Substance*

weighted_value

Protocol Output - The value weighted by the *component's mole fraction as determined from the 'full_substance*.

Type typing.Union[float, int, propertyestimator.utils.quantities.EstimatedQuantity, pint.quantity.build_quantity_class.<locals>.Quantity, propertyestimator.properties.properties.ParameterGradient]

execute (*directory, available_resources*)

Execute the protocol.

Protocols may be chained together by passing the output of previous protocols as input to the current one.

Parameters

- **directory** (*str*) – The directory to store output data in.
- **available_resources** (*ComputeResources*) – The resources available to execute on.

Returns The output of the execution.

Return type Dict[str, Any]

allow_merging

Protocol Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is `True`.

Type `bool`

apply_replicator (*replicator*, *template_values*, *template_index=-1*, *template_value=None*, *update_input_references=False*)

Applies a *ProtocolReplicator* to this protocol. This method should clone any protocols whose id contains the id of the replicator (in the format *\$(replicator.id)*).

Parameters

- **replicator** (*ProtocolReplicator*) – The replicator to apply.
- **template_values** (*list of Any*) – A list of the values which will be inserted into the newly replicated protocols.

This parameter is mutually exclusive with *template_index* and *template_value*

- **template_index** (*int, optional*) – A specific value which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_value*.

- **template_value** (*Any, optional*) – A specific index which should be used for any protocols flagged as to be replicated by the replicator. This option is mainly used when replicating children of an already replicated protocol.

This parameter is mutually exclusive with *template_values* and must be set along with a *template_index*.

- **update_input_references** (*bool*) – If true, any protocols which take their input from a protocol which was flagged for replication will be updated to take input from the actually replicated protocol. This should only be set to true if this protocol is not nested within a workflow or a protocol group.

This option cannot be used when a specific *template_index* or *template_value* is provided.

Returns A dictionary of references to all of the protocols which have been replicated, with keys of original protocol ids. Each value is comprised of a list of the replicated protocol ids, and their index into the *template_values* array.

Return type dict of ProtocolPath and list of tuple of ProtocolPath and int

can_merge (*other*, *path_replacements=None*)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*BaseProtocol*) – The protocol to compare against.
- **path_replacements** (*list of tuple of str, optional*) – Replacements to make in any value reference protocol paths before comparing for equality.

Returns True if the two protocols are safe to merge.

Return type `bool`

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

get_class_attribute (*reference_path*)

Returns one of this protocols, or any of its children's, attributes directly (rather than its value).

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the attribute to return.

Returns The class attribute.

Return type `object`

get_value (*reference_path*)

Returns the value of one of this protocols inputs / outputs.

Parameters **reference_path** (`ProtocolPath`) – The path pointing to the value to return.

Returns The value of the input / output

Return type `Any`

get_value_references (*input_path*)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *input_path*) takes its value from.

Notes

Currently this method only functions correctly for an input value which is either currently a `ProtocolPath`, or a *list / dict* which contains at least one `ProtocolPath`.

Parameters **input_path** (`propertyestimator.workflow.utils.ProtocolPath`) – The input value to check.

Returns A dictionary of the protocol paths that the input targeted by *input_path* depends upon.

Return type `dict of ProtocolPath and ProtocolPath`

property id

The unique id of this protocol.

Type `str`

merge (*other*)

Merges another `BaseProtocol` with this one. The id of this protocol will remain unchanged.

It is assumed that `can_merge` has already returned that these protocols are compatible to be merged together.

Parameters **other** (`BaseProtocol`) – The protocol to merge into this one.

Returns A map between any original protocol ids and their new merged values.

Return type `Dict[str, str]`

replace_protocol (*old_id*, *new_id*)

Finds each input which came from a given protocol and redirects it to instead take input from a new one.

Notes

This method is mainly intended to be used only when merging multiple protocols into one.

Parameters

- **old_id** (*str*) – The id of the old input protocol.

- **new_id** (*str*) – The id of the new input protocol.

property schema

A serializable schema for this object.

Type *ProtocolSchema*

set_uuid (*value*)

Store the uuid of the calculation this protocol belongs to

Parameters **value** (*str*) – The uuid of the parent calculation.

set_value (*reference_path, value*)

Sets the value of one of this protocols inputs.

Parameters

- **reference_path** (*ProtocolPath*) – The path pointing to the value to return.
- **value** (*Any*) – The value to set.

1.5.10 Workflow Construction Utilities

<i>BaseReweightingProtocols</i>	
<i>BaseSimulationProtocols</i>	
<i>generate_base_reweighting_protocols</i>	Constructs a set of protocols which, when combined in a workflow schema, may be executed to reweight a set of existing data to estimate a particular property.
<i>generate_base_simulation_protocols</i>	Constructs a set of protocols which, when combined in a workflow schema, may be executed to run a single simulation to estimate a particular property.
<i>generate_gradient_protocol_group</i>	Constructs a set of protocols which, when combined in a workflow schema, may be executed to reweight a set of existing data to estimate a particular property.

BaseReweightingProtocols

```
class propertyestimator.protocols.utils.BaseReweightingProtocols (unpack_stored_data,
                                                                    analy-
                                                                    sis_protocol,
                                                                    decorre-
                                                                    late_statistics,
                                                                    decorre-
                                                                    late_trajectory,
                                                                    concate-
                                                                    nate_trajectories,
                                                                    concate-
                                                                    nate_statistics,
                                                                    build_reference_system,
                                                                    re-
                                                                    duced_reference_potential,
                                                                    build_target_system,
                                                                    re-
                                                                    duced_target_potential,
                                                                    mbar_protocol)
```

`__init__()`
 Initialize self. See help(type(self)) for accurate signature.

Methods

<code>__init__</code>	Initialize self.
<code>count(value)</code>	
<code>index(value, [start, [stop]])</code>	Raises ValueError if the value is not present.

Attributes

<code>analysis_protocol</code>	Alias for field number 1
<code>build_reference_system</code>	Alias for field number 6
<code>build_target_system</code>	Alias for field number 8
<code>concatenate_statistics</code>	Alias for field number 5
<code>concatenate_trajectories</code>	Alias for field number 4
<code>decorrelate_statistics</code>	Alias for field number 2
<code>decorrelate_trajectory</code>	Alias for field number 3
<code>mbar_protocol</code>	Alias for field number 10
<code>reduced_reference_potential</code>	Alias for field number 7
<code>reduced_target_potential</code>	Alias for field number 9
<code>unpack_stored_data</code>	Alias for field number 0

property analysis_protocol
 Alias for field number 1

property build_reference_system
 Alias for field number 6

property build_target_system
 Alias for field number 8

property concatenate_statistics
 Alias for field number 5

property concatenate_trajectories
 Alias for field number 4

count (*value*) → integer – return number of occurrences of value

property decorrelate_statistics
 Alias for field number 2

property decorrelate_trajectory
 Alias for field number 3

index (*value*[, *start*[, *stop*]]) → integer – return first index of value.
 Raises ValueError if the value is not present.

property mbar_protocol
 Alias for field number 10

property reduced_reference_potential
 Alias for field number 7

property reduced_target_potential

Alias for field number 9

property unpack_stored_data

Alias for field number 0

BaseSimulationProtocols

```
class propertyestimator.protocols.utils.BaseSimulationProtocols (build_coordinates,
                                                                as-
                                                                sign_parameters,
                                                                en-
                                                                ergy_minimisation,
                                                                equilibra-
                                                                tion_simulation,
                                                                produc-
                                                                tion_simulation,
                                                                analy-
                                                                sis_protocol,
                                                                con-
                                                                verge_uncertainty,
                                                                ex-
                                                                tract_uncorrelated_trajectory,
                                                                ex-
                                                                tract_uncorrelated_statistics)
```

__init__ ()

Initialize self. See help(type(self)) for accurate signature.

Methods

<i>__init__</i>	Initialize self.
<i>count</i> (value)	
<i>index</i> (value, [start, [stop]])	Raises ValueError if the value is not present.

Attributes

<i>analysis_protocol</i>	Alias for field number 5
<i>assign_parameters</i>	Alias for field number 1
<i>build_coordinates</i>	Alias for field number 0
<i>converge_uncertainty</i>	Alias for field number 6
<i>energy_minimisation</i>	Alias for field number 2
<i>equilibration_simulation</i>	Alias for field number 3
<i>extract_uncorrelated_statistics</i>	Alias for field number 8
<i>extract_uncorrelated_trajectory</i>	Alias for field number 7
<i>production_simulation</i>	Alias for field number 4

property analysis_protocol

Alias for field number 5

property assign_parameters

Alias for field number 1

property build_coordinates

Alias for field number 0

property converge_uncertainty

Alias for field number 6

count (*value*) → integer – return number of occurrences of value

property energy_minimisation

Alias for field number 2

property equilibration_simulation

Alias for field number 3

property extract_uncorrelated_statistics

Alias for field number 8

property extract_uncorrelated_trajectory

Alias for field number 7

index (*value*[, *start*[, *stop*]]) → integer – return first index of value.

Raises ValueError if the value is not present.

property production_simulation

Alias for field number 4

propertyestimator.protocols.utils.generate_base_reweighting_protocols

propertyestimator.protocols.utils.**generate_base_reweighting_protocols** (*analysis_protocol*,
mbar_protocol,
work-
flow_options,
repli-
ca-
tor_id='data_repl',
id_suffix="")

Constructs a set of protocols which, when combined in a workflow schema, may be executed to reweight a set of existing data to estimate a particular property. The reweighted observable of interest will be calculated by following the passed in *analysis_protocol*.

Parameters

- **analysis_protocol** (*AveragePropertyProtocol*) – The protocol which will take input from the stored data, and generate a set of observables to reweight.
- **mbar_protocol** (*BaseReweightingProtocol*) – A template mbar reweighting protocol, which has it's reference observables already set. This method will automatically set the reduced potentials on this object.
- **workflow_options** (*WorkflowOptions*) – The options being used to generate a workflow.
- **replicator_id** (*str*) – The id to use for the data replicator.
- **id_suffix** (*str*) – A string suffix to append to each of the protocol ids.

Returns

- *BaseReweightingProtocols* – A named tuple of the protocol which should form the bulk of a property estimation workflow.

- *ProtocolReplicator* – A replicator which will clone the workflow for each piece of stored data.

propertyestimator.protocols.utils.generate_base_simulation_protocols

propertyestimator.protocols.utils.**generate_base_simulation_protocols** (*analysis_protocol*,
work-
flow_options,
id_suffix="",
condi-
tional_group=None)

Constructs a set of protocols which, when combined in a workflow schema, may be executed to run a single simulation to estimate a particular property. The observable of interest to extract from the simulation is determined by the passed in *analysis_protocol*.

The protocols returned will:

- 1) Build a set of liquid coordinates for the property substance using packmol.
- 2) Assign a set of smirnoff force field parameters to the system.
- 3) Perform an energy minimisation on the system.
- 4) Run a short NPT equilibration simulation for 100000 steps using a timestep of 2fs.
- 5) Within a conditional group (up to a maximum of 100 times):
 - 5a) Run a longer NPT production simulation for 1000000 steps using a timestep of 2fs
 - 5b) Extract the average value of an observable and it's uncertainty.
 - 5c) **If a convergence mode is set by the options, check if the target uncertainty has been met.**
If not, repeat steps 5a), 5b) and 5c).
- 6) Extract uncorrelated configurations from a generated production simulation.
- 7) Extract uncorrelated statistics from a generated production simulation.

Parameters

- **analysis_protocol** (*AveragePropertyProtocol*) – The protocol which will extract the observable of interest from the generated simulation data.
- **workflow_options** (*WorkflowOptions*) – The options being used to generate a workflow.
- **id_suffix** (*str*) – A string suffix to append to each of the protocol ids.
- **conditional_group** (*ProtocolGroup*, *optional*) – A custom group to wrap the main simulation / extraction protocols within. It is up to the caller of this method to manually add the convergence conditions to this group. If *None*, a default group with uncertainty convergence conditions is automatically constructed.

Returns

- *BaseSimulationProtocols* – A named tuple of the generated protocols.
- *ProtocolPath* – A reference to the final value of the estimated observable and its uncertainty (an *EstimatedQuantity*).
- *WorkflowSimulationDataToStore* – An object which describes the default data from a simulation to store, such as the uncorrelated statistics and configurations.

propertyestimator.protocols.utils.generate_gradient_protocol_group

`propertyestimator.protocols.utils.generate_gradient_protocol_group` (*template_reweighting_protocol*, *reference_force_field_paths*, *target_force_field_path*, *coordinate_file_path*, *trajectory_file_path*, *statistics_file_path*=", *replicator_id*='repl', *perturbation_scale*=0.0001, *substance_source*=None, *id_suffix*", *enable_pbc*=True, *use_subset_of_force_field*=True, *effective_sample_indices*=None)

Constructs a set of protocols which, when combined in a workflow schema, may be executed to reweight a set of existing data to estimate a particular property. The reweighted observable of interest will be calculated by following the passed in *analysis_protocol*.

Parameters

- **template_reweighting_protocol** (`BaseMBARProtocol`) – A template protocol which will be used to reweight the observable of interest to small perturbations to the parameter of interest. These will then be used to calculate the finite difference gradient. This utility takes care of setting the target and reference reduced potentials.

In the case that the template is of type *ReweightStatistics* and the observable is an energy, the statistics path will automatically be pointed to the energies evaluated using the perturbed parameter as opposed to the energy measured during the reference simulation.

- **reference_force_field_paths** (`ProtocolPath` or *list of ProtocolPath*) – The paths to the force field parameters which were used to generate the trajectories from which the observables of interest were calculated.
- **target_force_field_path** (`ProtocolPath`) –
The path to the force field parameters which the observables are being estimated at (this is mainly only useful when estimating the gradients of reweighted observables).
- **coordinate_file_path** (`ProtocolPath`) – A path to the initial coordinates of the simulation trajectory which was used to estimate the observable of interest.
- **trajectory_file_path** (`ProtocolPath`) – A path to the simulation trajectory which was used to estimate the observable of interest.
- **statistics_file_path** (`ProtocolPath`, *optional*) – A path to the statistics where were generated from the trajectory passed to the *trajectory_file_path* parameter. This is optional in cases where multiple reference force fields are passed to this method.

- **replicator_id** (*str*) – A unique id which will be used for the protocol replicator which will replicate this group for every parameter of interest.
- **perturbation_scale** (*float*) – The default amount to perturb parameters by.
- **substance_source** (*PlaceholderInput, optional*) – An optional protocol path to the substance whose gradient is being estimated. If None, the global property substance is used.
- **id_suffix** (*str*) – An optional string to append to the end of each of the protocol ids.
- **enable_pbc** (*bool*) – If true, periodic boundary conditions are employed when recalculating the reduced potentials.
- **use_subset_of_force_field** (*bool*) – If True, any reduced potentials will only be calculated from a subset of the force field which depends on the parameter of interest.
- **effective_sample_indices** (*ProtocolPath, optional*) – A placeholder variable which can be used to make the gradient protocols dependant on an MBAR protocol to ensure gradients aren't calculated when the MBAR protocol failed due to insufficient samples.

Returns

- *ProtocolGroup* – The protocol group which will estimate the gradient of an observable with respect to one parameter.
- *ProtocolReplicator* – The replicator which will copy the gradient group for every parameter of interest.
- *ProtocolPath* – A protocol path which points to the final gradient value.

1.6 Release History

Releases will eventually follow the `major.minor.micro` scheme recommended by [PEP440](#), where

- `major` increments denote a change that may break API compatibility with previous `major` releases
- `minor` increments add features but do not break API compatibility
- `micro` increments represent bugfix releases or improvements in documentation

All early releases however will simply receive a `micro` version bump regardless of how major the changes may be.

1.6.1 0.0.8 - ThermoML Improvements

This release is centered around cleaning up the ThermoML data set utilities. The main change is that ThermoML archive files can now be loaded even if they don't contain measurement uncertainties.

New Features

- PR #142: ThermoML archives without uncertainties can now be loaded.

Breaking Changes

- PR #142: All *ThermoMLXXX* classes other than *ThermoMLDataSet* are now private.

1.6.2 0.0.7 - Bug Quick Fixes

This release aims to fix a number of minor bugs.

Bugfixes

- PR #136: Fix for comparing thermodynamic states with unset pressures.
- PR #138: Fix for a typo in the maximum number of minimization iterations.

1.6.3 0.0.6 - Solvation Free Energies

This release centers around two key changes -

- i) a general refactoring of the protocol classes to be much cleaner and extensible through the removal of the old stub functions and the addition of cleaner descriptors.
- ii) the addition of workflows to estimate solvation free energies via the new `SolvationYankProtocol` and `SolvationFreeEnergy` classes.

The implemented free energy workflow is still rather basic, and does not yet support calculating parameter gradients or estimation from cached simulation data through reweighting.

A new table has been added to the documentation to make clear which built-in properties support which features.

New Features

- PR #110: Cleanup and refactor of protocol classes.
- PR #125: Support for PBS based HPC clusters.
- PR #127: Adds a basic workflow for estimating solvation free energies with `YANK`.
- PR #130: Adds a cleaner mechanism for restarting simulations from checkpoints.
- PR #134: Update to a more stable dask version.

Bugfixes

- PR #128: Removed the defunct dask backend `processes` kwarg.
- PR #133: Fix for tests failing on MacOS due to `travis` issues.

Breaking Changes

- PR #130: The `RunOpenMMSimulation.steps` input has now been split into the `steps_per_iteration` and `total_number_of_iterations` inputs.

Migration Guide

This release contained several public API breaking changes. For the most part, these can be remedied by the follow steps:

- Replace all instances of `run_openmm_simulation_protocol.steps` to `run_openmm_simulation_protocol.steps_per_iteration`

1.6.4 0.0.5 - Fix For Merging of Estimation Requests

This release implements a fix for a major bug which caused incorrect results to be returned when submitting multiple estimation requests at the same time - namely, the returned results became jumbled between the different requests. As an example, if a request was made to estimate a data set using the *smirnoff99frosst* force field, and then straight after with the *gaff 1.81* force field, the results of the *smirnoff99frosst* request may contain some properties estimated with *gaff 1.81* and vice versa.

This issue does not affect cases where only a single request was made and completed at a time (i.e the results of the previous request completed before the next estimation request was made).

Bugfixes

- PR #119: Fixes gather task merging.
- PR #121: Update to distributed 2.5.1.

1.6.5 0.0.4 - Initial Support for Non-SMIRNOFF FFs

This release adds initial support for estimating property data sets using force fields not based on the SMIRNOFF specification. In particular, initial AMBER force field support has been added, along with a protocol which applies said force fields using `tLeap`.

New Features

- PR #96: Adds a mechanism for specifying force fields not in the SMIRNOFF spec.
- PR #99: Adds support for applying AMBER force field parameters through `tLeap`
- PR #111: Protocols now stream trajectories from disk, rather than pre-load the whole thing.
- PR #112: Specific types of protocols can now be easily be replaced using `WorkflowOptions`.
- PR #117: Adds support for converting `PhysicalPropertyDataSet` objects to `pandas.DataFrame`.

Bugfixes

- PR #115: Fixes caching data for substances whose smiles contain forward slashes.
- PR #116: Fixes inconsistent mole fraction rounding.

Breaking Changes

- PR #96: The `PropertyEstimatorClient.request_estimate(force_field=...)` argument has been renamed to `force_field_source`.

Migration Guide

This release contained several public API breaking changes. For the most part, these can be remedied by the follow steps:

- Change all instances of `PropertyEstimatorClient.request_estimate(force_field=...)` to `PropertyEstimatorClient.request_estimate(force_field_source=...)`

1.6.6 0.0.3 - ExcessMolarVolume and Typing Improvements

This release implements a number of bug fixes and adds two key new features, namely built in support for estimating excess molar volume measurements, and improved type checking for protocol inputs and outputs.

New Features

- PR #98: Substance objects may now have components with multiple amount types.
- PR #101: Added support for estimating `ExcessMolarVolume` measurements from simulations.
- PR #104: `typing.Union` is now a valid type arguement to `protocol_output` and `protocol_input`.

Bugfixes

- PR #94: Fixes exception when testing equality of `ProtocolPath` objects.
- PR #100: Fixes precision issues when ensuring mole fractions are ≤ 1.0 .
- PR #102: Fixes replicated input for children of replicated protocols.
- PR #105: Fixes excess properties weighting by the wrong mole fractions.
- PR #107: Fixes excess properties being converged to the wrong uncertainty.
- PR #108: Fixes calculating MBAR gradients of reweighted properties.

Breaking Changes

- PR #98: `Substance.get_amount` renamed to `Substance.get_amounts` and now returns an immutable frozenset of `Amount` objects, rather than a single `Amount`.
- PR #104: The `DivideGradientByScalar`, `MultiplyGradientByScalar`, `AddGradients`, `SubtractGradients` and `WeightGradientByMoleFraction` protocols have been removed. The `WeightQuantityByMoleFraction` protocol has been renamed to `WeightByMoleFraction`.

Migration Guide

This release contained several public API breaking changes. For the most part, these can be remedied by the follow steps:

- Change all instances of `Substance.get_amount` to `Substance.get_amounts` and handle the newly returned frozenset of amounts, rather than the previously returned single amount.
- Replace the now removed protocols as follows:
 - `DivideGradientByScalar` -> `DivideValue`
 - `MultiplyGradientByScalar` -> `MultiplyValue`
 - `AddGradients` -> `AddValues`
 - `SubtractGradients` -> `SubtractValues`
 - `WeightGradientByMoleFraction` -> `WeightByMoleFraction`
 - `WeightQuantityByMoleFraction` -> `WeightByMoleFraction`

1.6.7 0.0.2 - Replicator Quick Fixes

A minor release to fix a number of minor bugs related to replicating protocols.

Bugfixes

- PR #90: Fixes merging gradient protocols with the same id.
- PR #92: Fixes replicating protocols for more than 10 template values.
- PR #93: Fixes ConditionalGroup objects losing their conditions input.

1.6.8 0.0.1 - Initial Release

The initial pre-alpha release of the framework.

1.7 Release Process

This document aims to outline the steps needed to release the `propertyestimator` on `omnia`. This should only be done with the approval of the core maintainers.

1.7.1 1. Update the Release History

If no PR has been submitted, create a new one to keep track of changes to the release notes *only*. Only the `releasehistory.rst` file may be edited in this PR.

Ensure that the release history file is up to date, and conforms to the below template:

```
X.Y.Z - Descriptive Title
-----

This release...

New Features
*****

* PR #X: Feature summary

Bugfixes
*****

* PR #Y: Fix Summary

Breaking Changes
*****

* PR #Z: Descriptive summary of the breaking change

Migration Guide
*****

This release contained several public API breaking changes. For the most part, these ↵
↵ can be
```

(continues on next page)

```
remedied by the follow steps:  
  
* A somewhat verbose guide on how users should upgrade their code given the new_  
↳breaking changes.
```

1.7.2 2: Cut the Release on GitHub

To cut a new release on GitHub:

- 1) Go to the Releases tab on the front page of the repo and choose Create a new release.
- 2) Set the release tag using the form: X.Y.Z
- 3) Added a descriptive title using the form: X.Y.Z [Descriptive Title]
- 4) Ensure the This is a pre-release checkbox is ticked.
- 5) Reformat the release notes from part 1) into markdown and paste into the description box.
 - a) Append the following extra message above the *New Features* title:

```
A richer version of these release notes with live links to API documentation is_  
↳available  
on [our ReadTheDocs page](https://property-estimator.readthedocs.io/en/latest/  
↳releasehistory.html)  
  
See our [installation instructions](https://property-estimator.readthedocs.io/en/  
↳latest/install.html) .  
  
Please report bugs, request features, or ask questions through our  
[issue tracker](https://github.com/openforcefield/propertyestimator/issues) .  
  
**Please note that this is a pre-alpha release and there will still be major changes_  
↳to the API  
prior to a stable 1.0.0 release.**
```

Note - You do not need to upload any files. The source code will automatically be added as a '.tar.gz' file.

1.7.3 3: Trigger a New Build on Omnia

To trigger the build in omnia:

- 1) Create branch or fork of omnia-md/conda-recipes with the following changes to propertyestimator in meta.yaml:
 - a) Set git_tag to match the git release tag
 - b) Update the version to match the release (this will go into the conda package name)
 - c) Set build to 0
 - d) Update any dependencies in the requirements section
 - e) If we want to push to special rc label use extra.upload
- 2) Open PR to merge branch or fork into omnia-md master:
 - a) The PR title should have the format [propertyestimator] X.Y.Z (label: rc)
 - b) No PR body text is needed

- c) Travis will run on this PR (~30 minutes) and attempt to build the package. Under no conditions will the package be uploaded before the PR is merged. This step is just to ensure that building doesn't crash.
 - d) If the build is successful the PR should be reviewed and merged by the `omnia` maintainers
 - e) **Once merged into master** the package is built again on travis, and pushed to the channel set in `meta.yaml` (`main`, `beta`, or `rc`)
- 3) Test the `omnia` package:
- a) `conda install -c omnia/label/rc propertyestimator`

Note: Omnia builds take about 30 minutes to run. When you open a PR the build will run, and you can check the bottom of the travis logs for "package failed to build" listings. Some packages always fail (protons, assaytools), but propertyestimator shouldn't be there. Ctrl-F for "propertyestimator" to ensure that it did build at all though.

1.7.4 4: Update the ReadTheDocs Build Versions

To ensure that the read the docs pages are updated:

- 1) Trigger a RTD build of `latest`.
- 2) Under the `Versions` tab add the new release version to the list of built versions and **save**.
- 3) Verify the new version docs have been built and pushed correctly
- 4) Under `Admin | Advanced Settings`: Set the new release version as `Default version to display` and **save**.

Symbols

<code>__init__()</code>	(<i>propertyestimator.backends.BaseDaskBackend</i> method), 69	<code>__init__()</code>	(<i>propertyestimator.forcefield.LigParGenForceFieldSource</i> method), 23
<code>__init__()</code>	(<i>propertyestimator.backends.ComputeResources</i> method), 67	<code>__init__()</code>	(<i>propertyestimator.forcefield.SmirnoffForceFieldSource</i> method), 20
<code>__init__()</code>	(<i>propertyestimator.backends.DaskLSFBackend</i> method), 71	<code>__init__()</code>	(<i>propertyestimator.forcefield.TLeapForceFieldSource</i> method), 21
<code>__init__()</code>	(<i>propertyestimator.backends.DaskLocalCluster</i> method), 70	<code>__init__()</code>	(<i>propertyestimator.layers.PropertyCalculationLayer</i> method), 63
<code>__init__()</code>	(<i>propertyestimator.backends.PropertyEstimatorBackend</i> method), 66	<code>__init__()</code>	(<i>propertyestimator.layers.ReweightingLayer</i> method), 64
<code>__init__()</code>	(<i>propertyestimator.backends.QueueWorkerResources</i> method), 68	<code>__init__()</code>	(<i>propertyestimator.layers.SimulationLayer</i> method), 65
<code>__init__()</code>	(<i>propertyestimator.client.ConnectionOptions</i> method), 18	<code>__init__()</code>	(<i>propertyestimator.properties.CalculationSource</i> method), 32
<code>__init__()</code>	(<i>propertyestimator.client.PropertyEstimatorClient</i> method), 12	<code>__init__()</code>	(<i>propertyestimator.properties.Density</i> method), 33
<code>__init__()</code>	(<i>propertyestimator.client.PropertyEstimatorOptions</i> method), 14	<code>__init__()</code>	(<i>propertyestimator.properties.DielectricConstant</i> method), 37
<code>__init__()</code>	(<i>propertyestimator.client.PropertyEstimatorResult</i> method), 17	<code>__init__()</code>	(<i>propertyestimator.properties.EnthalpyOfMixing</i> method), 40
<code>__init__()</code>	(<i>propertyestimator.client.PropertyEstimatorSubmission</i> method), 16	<code>__init__()</code>	(<i>propertyestimator.properties.EnthalpyOfVaporization</i> method), 42
<code>__init__()</code>	(<i>propertyestimator.datasets.PhysicalPropertyDataSet</i> method), 53	<code>__init__()</code>	(<i>propertyestimator.properties.ExcessMolarVolume</i> method), 35
<code>__init__()</code>	(<i>propertyestimator.datasets.ThermoMLDataSet</i> method), 57	<code>__init__()</code>	(<i>propertyestimator.properties.HostGuestBindingAffinity</i> method), 44
<code>__init__()</code>	(<i>propertyestimator.forcefield.ForceFieldSource</i> method), 19	<code>__init__()</code>	(<i>propertyestimator.properties.MeasurementSource</i> method), 31
		<code>__init__()</code>	(<i>propertyestimator.properties.ParameterGradient</i> method), 25

<code>__init__()</code> (propertyestimator.properties.ParameterGradientKey method), 24	<code>__init__()</code> (propertyestimator.protocols.miscellaneous.AddValues method), 224
<code>__init__()</code> (propertyestimator.properties.PhysicalProperty method), 28	<code>__init__()</code> (propertyestimator.protocols.miscellaneous.DivideValue method), 235
<code>__init__()</code> (propertyestimator.properties.PropertyPhase method), 30	<code>__init__()</code> (propertyestimator.protocols.miscellaneous.FilterSubstanceByRole method), 239
<code>__init__()</code> (propertyestimator.properties.Source method), 30	<code>__init__()</code> (propertyestimator.protocols.miscellaneous.MultiplyValue method), 231
<code>__init__()</code> (propertyestimator.protocols.analysis.AveragePropertyProtocol method), 149	<code>__init__()</code> (propertyestimator.protocols.miscellaneous.SubtractValues method), 228
<code>__init__()</code> (propertyestimator.protocols.analysis.AverageTrajectoryProperty method), 153	<code>__init__()</code> (propertyestimator.protocols.miscellaneous.WeightByMoleFraction method), 243
<code>__init__()</code> (propertyestimator.protocols.analysis.ExtractAverageStatistic method), 158	<code>__init__()</code> (propertyestimator.protocols.reweighting.BaseMBARProtocol method), 187
<code>__init__()</code> (propertyestimator.protocols.analysis.ExtractUncorrelatedData method), 162	<code>__init__()</code> (propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM method), 182
<code>__init__()</code> (propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData method), 170	<code>__init__()</code> (propertyestimator.protocols.reweighting.ConcatenateStatistics method), 178
<code>__init__()</code> (propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData method), 166	<code>__init__()</code> (propertyestimator.protocols.reweighting.ConcatenateTrajectories method), 174
<code>__init__()</code> (propertyestimator.protocols.coordinates.BuildCoordinatesPackmol method), 99	<code>__init__()</code> (propertyestimator.protocols.reweighting.ReweightStatistics method), 191
<code>__init__()</code> (propertyestimator.protocols.coordinates.BuildDockedCoordinates method), 108	<code>__init__()</code> (propertyestimator.protocols.simulation.RunEnergyMinimisation method), 122
<code>__init__()</code> (propertyestimator.protocols.coordinates.SolvateExistingStructure method), 103	<code>__init__()</code> (propertyestimator.protocols.simulation.RunOpenMMSimulation method), 126
<code>__init__()</code> (propertyestimator.protocols.forcefield.BuildSmirnoffSystem method), 113	<code>__init__()</code> (propertyestimator.protocols.storage.UnpackStoredDataCollection method), 215
<code>__init__()</code> (propertyestimator.protocols.forcefield.BuildTLeapSystem method), 117	<code>__init__()</code> (propertyestimator.protocols.storage.UnpackStoredSimulationData method), 219
<code>__init__()</code> (propertyestimator.protocols.gradients.CentralDifferenceGradient method), 203	<code>__init__()</code> (propertyestimator.protocols.utils.BaseReweightingProtocols method), 247
<code>__init__()</code> (propertyestimator.protocols.gradients.GradientReducedPotentials method), 197	<code>__init__()</code> (propertyestimator.protocols.utils.BaseSimulationProtocols method), 249
<code>__init__()</code> (propertyestimator.protocols.groups.ConditionalGroup method), 211	<code>__init__()</code> (propertyestimator.protocols.yank.BaseYankProtocol method),
<code>__init__()</code> (propertyestimator.protocols.groups.ProtocolGroup method),	

132			
<code>__init__()</code>	(<i>propertyestimator.protocols.yank.LigandReceptorYankProtocol</i> method), 136	<code>__init__()</code>	(<i>propertyestimator.workflow.schemas.ProtocolReplicator</i> method), 87
<code>__init__()</code>	(<i>propertyestimator.protocols.yank.SolvationYankProtocol</i> method), 143	<code>__init__()</code>	(<i>propertyestimator.workflow.schemas.ProtocolSchema</i> method), 85
<code>__init__()</code>	(<i>propertyestimator.server.PropertyEstimatorServer</i> method), 26	<code>__init__()</code>	(<i>propertyestimator.workflow.schemas.WorkflowDataCollectionToStore</i> method), 90
<code>__init__()</code>	(<i>propertyestimator.storage.LocalFileStorage</i> method), 75	<code>__init__()</code>	(<i>propertyestimator.workflow.schemas.WorkflowOutputToStore</i> method), 89
<code>__init__()</code>	(<i>propertyestimator.storage.PropertyEstimatorStorage</i> method), 73	<code>__init__()</code>	(<i>propertyestimator.workflow.schemas.WorkflowSchema</i> method), 84
<code>__init__()</code>	(<i>propertyestimator.storage.dataclasses.BaseStoredData</i> method), 77	<code>__init__()</code>	(<i>propertyestimator.workflow.schemas.WorkflowSimulationDataToStore</i> method), 90
<code>__init__()</code>	(<i>propertyestimator.storage.dataclasses.StoredDataCollection</i> method), 80	<code>__init__()</code>	(<i>propertyestimator.workflow.utils.PlaceholderInput</i> method), 94
<code>__init__()</code>	(<i>propertyestimator.storage.dataclasses.StoredSimulationData</i> method), 79	<code>__init__()</code>	(<i>propertyestimator.workflow.utils.ProtocolPath</i> method), 95
<code>__init__()</code>	(<i>propertyestimator.substances.Substance</i> method), 47	<code>__init__()</code>	(<i>propertyestimator.workflow.utils.ReplicatorValue</i> method), 95
<code>__init__()</code>	(<i>propertyestimator.thermodynamics.ThermodynamicState</i> method), 51		
<code>__init__()</code>	(<i>propertyestimator.workflow.IWorkflowProperty</i> method), 84	A	
<code>__init__()</code>	(<i>propertyestimator.workflow.Workflow</i> method), 81	<code>activate_site_location</code>	(<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates</i> attribute), 110
<code>__init__()</code>	(<i>propertyestimator.workflow.WorkflowGraph</i> method), 82	<code>add_component()</code>	(<i>propertyestimator.substances.Substance</i> method), 50
<code>__init__()</code>	(<i>propertyestimator.workflow.WorkflowOptions</i> method), 83	<code>add_condition()</code>	(<i>propertyestimator.protocols.groups.ConditionalGroup</i> method), 213
<code>__init__()</code>	(<i>propertyestimator.workflow.decorators.BaseProtocolAttribute</i> method), 97	<code>add_socket()</code>	(<i>propertyestimator.server.PropertyEstimatorServer</i> method), 27
<code>__init__()</code>	(<i>propertyestimator.workflow.decorators.InequalityMergeBehaviour</i> method), 98	<code>add_sockets()</code>	(<i>propertyestimator.server.PropertyEstimatorServer</i> method), 27
<code>__init__()</code>	(<i>propertyestimator.workflow.decorators.MergeBehaviour</i> method), 98	<code>add_workflow()</code>	(<i>propertyestimator.workflow.WorkflowGraph</i> method), 83
<code>__init__()</code>	(<i>propertyestimator.workflow.protocols.BaseProtocol</i> method), 91	<code>AddValues</code>	(class in <i>propertyestimator.protocols.miscellaneous</i>), 224
<code>__init__()</code>	(<i>propertyestimator.workflow.schemas.ProtocolGroupSchema</i> method), 86	<code>allow_gpu_platforms</code>	(<i>propertyestimator.protocols.simulation.RunOpenMMSimulation</i> attribute), 128
		<code>allow_merging</code>	(<i>propertyestimator.protocols.analysis.AveragePropertyProtocol</i> attribute), 151
		<code>allow_merging</code>	(<i>propertyestimator</i> attribute), 151

	<i>tor.protocols.analysis.AverageTrajectoryProperty attribute</i>), 155		<i>tor.protocols.miscellaneous.SubtractValues attribute</i>), 229
allow_merging	(<i>propertyestimator.protocols.analysis.ExtractAverageStatistic attribute</i>), 159	allow_merging	(<i>propertyestimator.protocols.miscellaneous.WeightByMoleFraction attribute</i>), 244
allow_merging	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedData attribute</i>), 164	allow_merging	(<i>propertyestimator.protocols.reweighting.BaseMBARProtocol attribute</i>), 189
allow_merging	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData attribute</i>), 172	allow_merging	(<i>propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM attribute</i>), 184
allow_merging	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData attribute</i>), 168	allow_merging	(<i>propertyestimator.protocols.reweighting.ConcatenateStatistics attribute</i>), 180
allow_merging	(<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol attribute</i>), 101	allow_merging	(<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories attribute</i>), 176
allow_merging	(<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates attribute</i>), 110	allow_merging	(<i>propertyestimator.protocols.reweighting.ReweightStatistics attribute</i>), 193
allow_merging	(<i>propertyestimator.protocols.coordinates.SolvateExistingStructure attribute</i>), 105	allow_merging	(<i>propertyestimator.protocols.simulation.RunEnergyMinimisation attribute</i>), 124
allow_merging	(<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem attribute</i>), 114	allow_merging	(<i>propertyestimator.protocols.simulation.RunOpenMMSimulation attribute</i>), 129
allow_merging	(<i>propertyestimator.protocols.forcefield.BuildTLeapSystem attribute</i>), 119	allow_merging	(<i>propertyestimator.protocols.storage.UnpackStoredDataCollection attribute</i>), 217
allow_merging	(<i>propertyestimator.protocols.gradients.CentralDifferenceGradient attribute</i>), 204	allow_merging	(<i>propertyestimator.protocols.storage.UnpackStoredSimulationData attribute</i>), 221
allow_merging	(<i>propertyestimator.protocols.gradients.GradientReducedPotentials attribute</i>), 200	allow_merging	(<i>propertyestimator.protocols.yank.BaseYankProtocol attribute</i>), 134
allow_merging	(<i>propertyestimator.protocols.groups.ConditionalGroup attribute</i>), 214	allow_merging	(<i>propertyestimator.protocols.yank.LigandReceptorYankProtocol attribute</i>), 139
allow_merging	(<i>propertyestimator.protocols.groups.ProtocolGroup attribute</i>), 210	allow_merging	(<i>propertyestimator.protocols.yank.SolvationYankProtocol attribute</i>), 146
allow_merging	(<i>propertyestimator.protocols.miscellaneous.AddValues attribute</i>), 225	allow_merging	(<i>propertyestimator.workflow.protocols.BaseProtocol attribute</i>), 92
allow_merging	(<i>propertyestimator.protocols.miscellaneous.DivideValue attribute</i>), 237	allow_protocol_merging	(<i>propertyestimator.client.PropertyEstimatorOptions attribute</i>), 14
allow_merging	(<i>propertyestimator.protocols.miscellaneous.FilterSubstanceByRole attribute</i>), 240	allowed_calculation_layers	(<i>propertyestimator.client.PropertyEstimatorOptions attribute</i>), 14
allow_merging	(<i>propertyestimator.protocols.miscellaneous.MultiplyValue attribute</i>), 233	analysis_protocol()	(<i>propertyestimator.protocols.utils.BaseReweightingProtocols property</i>), 248
allow_merging	(<i>propertyestimator</i>	analysis_protocol()	(<i>propertyestimator</i>

<i>tor.protocols.utils.BaseSimulationProtocols</i> <i>property</i>), 249	<i>tor.protocols.groups.ProtocolGroup</i> method), 210
<code>append_uuid()</code> (<i>propertyestimator.workflow.utils.ProtocolPath</i> method), 96	<code>apply_replicator()</code> (<i>propertyestimator.protocols.miscellaneous.AddValues</i> method), 225
<code>apply()</code> (<i>propertyestimator.workflow.schemas.ProtocolReplicator</i> method), 88	<code>apply_replicator()</code> (<i>propertyestimator.protocols.miscellaneous.DivideValue</i> method), 237
<code>apply_known_charges</code> (<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem</i> attribute), 114	<code>apply_replicator()</code> (<i>propertyestimator.protocols.miscellaneous.FilterSubstanceByRole</i> method), 241
<code>apply_replicator()</code> (<i>propertyestimator.protocols.analysis.AveragePropertyProtocol</i> method), 151	<code>apply_replicator()</code> (<i>propertyestimator.protocols.miscellaneous.MultiplyValue</i> method), 233
<code>apply_replicator()</code> (<i>propertyestimator.protocols.analysis.AverageTrajectoryProperty</i> method), 155	<code>apply_replicator()</code> (<i>propertyestimator.protocols.miscellaneous.SubtractValues</i> method), 229
<code>apply_replicator()</code> (<i>propertyestimator.protocols.analysis.ExtractAverageStatistic</i> method), 159	<code>apply_replicator()</code> (<i>propertyestimator.protocols.miscellaneous.WeightByMoleFraction</i> method), 245
<code>apply_replicator()</code> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedData</i> method), 164	<code>apply_replicator()</code> (<i>propertyestimator.protocols.reweighting.BaseMBARProtocol</i> method), 189
<code>apply_replicator()</code> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData</i> method), 172	<code>apply_replicator()</code> (<i>propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM</i> method), 184
<code>apply_replicator()</code> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData</i> method), 168	<code>apply_replicator()</code> (<i>propertyestimator.protocols.reweighting.ConcatenateStatistics</i> method), 180
<code>apply_replicator()</code> (<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol</i> method), 101	<code>apply_replicator()</code> (<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories</i> method), 176
<code>apply_replicator()</code> (<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates</i> method), 110	<code>apply_replicator()</code> (<i>propertyestimator.protocols.reweighting.ReweightStatistics</i> method), 193
<code>apply_replicator()</code> (<i>propertyestimator.protocols.coordinates.SolvateExistingStructure</i> method), 105	<code>apply_replicator()</code> (<i>propertyestimator.protocols.simulation.RunEnergyMinimisation</i> method), 124
<code>apply_replicator()</code> (<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem</i> method), 114	<code>apply_replicator()</code> (<i>propertyestimator.protocols.simulation.RunOpenMMSimulation</i> method), 129
<code>apply_replicator()</code> (<i>propertyestimator.protocols.forcefield.BuildTLeapSystem</i> method), 119	<code>apply_replicator()</code> (<i>propertyestimator.protocols.storage.UnpackStoredDataCollection</i> method), 217
<code>apply_replicator()</code> (<i>propertyestimator.protocols.gradients.CentralDifferenceGradient</i> method), 205	<code>apply_replicator()</code> (<i>propertyestimator.protocols.storage.UnpackStoredSimulationData</i> method), 221
<code>apply_replicator()</code> (<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> method), 200	<code>apply_replicator()</code> (<i>propertyestimator.protocols.yank.BaseYankProtocol</i> method), 134
<code>apply_replicator()</code> (<i>propertyestimator.protocols.groups.ConditionalGroup</i> method), 214	<code>apply_replicator()</code> (<i>propertyestimator.protocols.yank.LigandReceptorYankProtocol</i> method), 139
<code>apply_replicator()</code> (<i>propertyestima-</i>	<code>apply_replicator()</code> (<i>propertyestima-</i>

- tor.protocols.yank.SolvationYankProtocol* method), 146
- `apply_replicator()` (*propertyestimator.workflow.protocols.BaseProtocol* method), 93
- `apply_restraints` (*propertyestimator.protocols.yank.LigandReceptorYankProtocol* attribute), 139
- `assign_parameters()` (*propertyestimator.protocols.utils.BaseSimulationProtocols* property), 249
- `AveragePropertyProtocol` (class in *propertyestimator.protocols.analysis*), 149
- `AverageTrajectoryProperty` (class in *propertyestimator.protocols.analysis*), 153
- ## B
- `BaseDaskBackend` (class in *propertyestimator.backends*), 69
- `BaseMBARProtocol` (class in *propertyestimator.protocols.reweighting*), 187
- `BaseProtocol` (class in *propertyestimator.workflow.protocols*), 90
- `BaseProtocolAttribute` (class in *propertyestimator.workflow.decorators*), 97
- `BaseReweightingProtocols` (class in *propertyestimator.protocols.utils*), 247
- `BaseSimulationProtocols` (class in *propertyestimator.protocols.utils*), 249
- `BaseStoredData` (class in *propertyestimator.storage.dataclasses*), 77
- `BaseYankProtocol` (class in *propertyestimator.protocols.yank*), 132
- `beta()` (*propertyestimator.thermodynamics.ThermodynamicState* property), 52
- `bind()` (*propertyestimator.server.PropertyEstimatorServer* method), 27
- `bootstrap_iterations` (*propertyestimator.protocols.analysis.AveragePropertyProtocol* attribute), 150
- `bootstrap_iterations` (*propertyestimator.protocols.analysis.AverageTrajectoryProperty* attribute), 156
- `bootstrap_iterations` (*propertyestimator.protocols.analysis.ExtractAverageStatistic* attribute), 160
- `bootstrap_iterations` (*propertyestimator.protocols.reweighting.BaseMBARProtocol* attribute), 188
- `bootstrap_iterations` (*propertyestimator.protocols.reweighting.ReweightStatistics* attribute), 194
- `bootstrap_sample_size` (*propertyestimator.protocols.analysis.AveragePropertyProtocol* attribute), 150
- `bootstrap_sample_size` (*propertyestimator.protocols.analysis.AverageTrajectoryProperty* attribute), 156
- `bootstrap_sample_size` (*propertyestimator.protocols.analysis.ExtractAverageStatistic* attribute), 160
- `bootstrap_sample_size` (*propertyestimator.protocols.reweighting.BaseMBARProtocol* attribute), 188
- `bootstrap_sample_size` (*propertyestimator.protocols.reweighting.ReweightStatistics* attribute), 194
- `bootstrap_uncertainties` (*propertyestimator.protocols.reweighting.BaseMBARProtocol* attribute), 188
- `bootstrap_uncertainties` (*propertyestimator.protocols.reweighting.ReweightStatistics* attribute), 194
- `box_aspect_ratio` (*propertyestimator.protocols.coordinates.BuildCoordinatesPackmol* attribute), 100
- `box_aspect_ratio` (*propertyestimator.protocols.coordinates.SolvateExistingStructure* attribute), 105
- `build_coordinates()` (*propertyestimator.protocols.utils.BaseSimulationProtocols* property), 250
- `build_reference_system()` (*propertyestimator.protocols.utils.BaseReweightingProtocols* property), 248
- `build_target_system()` (*propertyestimator.protocols.utils.BaseReweightingProtocols* property), 248
- `BuildCoordinatesPackmol` (class in *propertyestimator.protocols.coordinates*), 99
- `BuildDockedCoordinates` (class in *propertyestimator.protocols.coordinates*), 108
- `BuildDockedCoordinates.ActivateSiteLocation` (class in *propertyestimator.protocols.coordinates*), 109
- `BuildSmirnoffSystem` (class in *propertyestimator.protocols.forcefield*), 113
- `BuildSmirnoffSystem.WaterModel` (class in *propertyestimator.protocols.forcefield*), 114
- `BuildTLeapSystem` (class in *propertyestimator.protocols.forcefield*), 117
- `BuildTLeapSystem.ChargeBackend` (class in *propertyestimator.protocols.forcefield*), 118
- `BuildTLeapSystem.WaterModel` (class in *propertyestimator.protocols.forcefield*), 119

C

<code>calculate_aqueous_ionic_mole_fraction()</code> (<i>propertyestimator.substances.Substance</i> static method), 50	<code>tor.protocols.miscellaneous.AddValues</code> method), 226
<code>CalculateReducedPotentialOpenMM</code> (class in <i>propertyestimator.protocols.reweighting</i>), 182	<code>can_merge()</code> (<i>propertyestimator.protocols.miscellaneous.DivideValue</i> method), 237
<code>CalculationSource</code> (class in <i>propertyestimator.properties</i>), 32	<code>can_merge()</code> (<i>propertyestimator.protocols.miscellaneous.FilterSubstanceByRole</i> method), 241
<code>can_merge()</code> (<i>propertyestimator.protocols.analysis.AveragePropertyProtocol</i> method), 152	<code>can_merge()</code> (<i>propertyestimator.protocols.miscellaneous.MultiplyValue</i> method), 234
<code>can_merge()</code> (<i>propertyestimator.protocols.analysis.AverageTrajectoryProperty</i> method), 156	<code>can_merge()</code> (<i>propertyestimator.protocols.miscellaneous.SubtractValues</i> method), 230
<code>can_merge()</code> (<i>propertyestimator.protocols.analysis.ExtractAverageStatistic</i> method), 160	<code>can_merge()</code> (<i>propertyestimator.protocols.miscellaneous.WeightByMoleFraction</i> method), 245
<code>can_merge()</code> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedData</i> method), 164	<code>can_merge()</code> (<i>propertyestimator.protocols.reweighting.BaseMBARProtocol</i> method), 190
<code>can_merge()</code> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData</i> method), 172	<code>can_merge()</code> (<i>propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM</i> method), 185
<code>can_merge()</code> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData</i> method), 168	<code>can_merge()</code> (<i>propertyestimator.protocols.reweighting.ConcatenateStatistics</i> method), 180
<code>can_merge()</code> (<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol</i> method), 102	<code>can_merge()</code> (<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories</i> method), 177
<code>can_merge()</code> (<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates</i> method), 111	<code>can_merge()</code> (<i>propertyestimator.protocols.reweighting.ReweightStatistics</i> method), 194
<code>can_merge()</code> (<i>propertyestimator.protocols.coordinates.SolvateExistingStructure</i> method), 106	<code>can_merge()</code> (<i>propertyestimator.protocols.simulation.RunEnergyMinimisation</i> method), 124
<code>can_merge()</code> (<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem</i> method), 115	<code>can_merge()</code> (<i>propertyestimator.protocols.simulation.RunOpenMMSimulation</i> method), 130
<code>can_merge()</code> (<i>propertyestimator.protocols.forcefield.BuildTLeapSystem</i> method), 119	<code>can_merge()</code> (<i>propertyestimator.protocols.storage.UnpackStoredDataCollection</i> method), 217
<code>can_merge()</code> (<i>propertyestimator.protocols.gradients.CentralDifferenceGradient</i> method), 205	<code>can_merge()</code> (<i>propertyestimator.protocols.storage.UnpackStoredSimulationData</i> method), 222
<code>can_merge()</code> (<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> method), 201	<code>can_merge()</code> (<i>propertyestimator.protocols.yank.BaseYankProtocol</i> method), 135
<code>can_merge()</code> (<i>propertyestimator.protocols.groups.ConditionalGroup</i> method), 213	<code>can_merge()</code> (<i>propertyestimator.protocols.yank.LigandReceptorYankProtocol</i> method), 140
<code>can_merge()</code> (<i>propertyestimator.protocols.groups.ProtocolGroup</i> method), 209	<code>can_merge()</code> (<i>propertyestimator.protocols.yank.SolvationYankProtocol</i> method), 146
<code>can_merge()</code> (<i>propertyestimator</i>	<code>can_merge()</code> (<i>propertyestimator</i>

<i>tor.storage.dataclasses.BaseStoredData</i> method), 78	ConcatenateTrajectories (class in <i>propertyestimator.protocols.reweighting</i>), 174
<code>can_merge()</code> (<i>propertyestimator.storage.dataclasses.StoredDataCollection</i> method), 80	ConditionalGroup (class in <i>propertyestimator.protocols.groups</i>), 211
<code>can_merge()</code> (<i>propertyestimator.storage.dataclasses.StoredSimulationData</i> method), 79	ConditionalGroup.ConditionType (class in <i>propertyestimator.protocols.groups</i>), 212
<code>can_merge()</code> (<i>propertyestimator.workflow.protocols.BaseProtocol</i> method), 92	ConnectionOptions (class in <i>propertyestimator.client</i>), 18
CentralDifferenceGradient (class in <i>propertyestimator.protocols.gradients</i>), 203	<code>converge_uncertainty()</code> (<i>propertyestimator.protocols.utils.BaseSimulationProtocols</i> property), 250
<code>charge_backend</code> (<i>propertyestimator.protocols.forcefield.BuildTLeapSystem</i> attribute), 118	<code>coordinate_file_name</code> (<i>propertyestimator.storage.dataclasses.StoredSimulationData</i> attribute), 78
<code>charged_molecule_paths</code> (<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem</i> attribute), 114	<code>coordinate_file_path</code> (<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol</i> attribute), 101
<code>checkpoint_frequency</code> (<i>propertyestimator.protocols.simulation.RunOpenMMSimulation</i> attribute), 128	<code>coordinate_file_path</code> (<i>propertyestimator.protocols.coordinates.SolvateExistingStructure</i> attribute), 106
<code>checkpoint_interval</code> (<i>propertyestimator.protocols.yank.BaseYankProtocol</i> attribute), 133	<code>coordinate_file_path</code> (<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem</i> attribute), 115
<code>checkpoint_interval</code> (<i>propertyestimator.protocols.yank.LigandReceptorYankProtocol</i> attribute), 140	<code>coordinate_file_path</code> (<i>propertyestimator.protocols.forcefield.BuildTLeapSystem</i> attribute), 120
<code>checkpoint_interval</code> (<i>propertyestimator.protocols.yank.SolvationYankProtocol</i> attribute), 147	<code>coordinate_file_path</code> (<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> attribute), 199
<code>collection_data_paths</code> (<i>propertyestimator.protocols.storage.UnpackStoredDataCollection</i> attribute), 216	<code>coordinate_file_path</code> (<i>propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM</i> attribute), 183
<code>component</code> (<i>propertyestimator.protocols.miscellaneous.WeightByMoleFraction</i> attribute), 244	<code>coordinate_file_path</code> (<i>propertyestimator.protocols.storage.UnpackStoredSimulationData</i> attribute), 221
<code>component_role</code> (<i>propertyestimator.protocols.miscellaneous.FilterSubstanceByRole</i> attribute), 240	<code>coordinate_file_path</code> (<i>propertyestimator.workflow.schemas.WorkflowSimulationDataToStore</i> attribute), 89
<code>components()</code> (<i>propertyestimator.substances.Substance</i> property), 49	<code>count()</code> (<i>propertyestimator.protocols.utils.BaseReweightingProtocols</i> method), 248
ComputeResources (class in <i>propertyestimator.backends</i>), 67	<code>count()</code> (<i>propertyestimator.protocols.utils.BaseSimulationProtocols</i> method), 250
ComputeResources.GPUSoftware (class in <i>propertyestimator.backends</i>), 67	<code>current_iteration</code> (<i>propertyestimator.protocols.groups.ConditionalGroup</i> attribute), 212
<code>concatenate_statistics()</code> (<i>propertyestimator.protocols.utils.BaseReweightingProtocols</i> property), 248	<code>cutoff()</code> (<i>propertyestimator.forcefield.LigParGenForceFieldSource</i> property), 23
<code>concatenate_trajectories()</code> (<i>propertyestimator.protocols.utils.BaseReweightingProtocols</i> property), 248	<code>cutoff()</code> (<i>propertyestimator.forcefield.TLeapForceFieldSource</i> property), 22
ConcatenateStatistics (class in <i>propertyestimator.protocols.reweighting</i>), 178	

D

DaskLocalCluster (class in <i>propertyestimator.backends</i>), 70	<i>dependencies()</i> (<i>propertyestimator.protocols.forcefield.BuildTLeapSystem property</i>), 120
DaskLSFBackend (class in <i>propertyestimator.backends</i>), 71	<i>dependencies()</i> (<i>propertyestimator.protocols.gradients.CentralDifferenceGradient property</i>), 205
data (<i>propertyestimator.storage.dataclasses.StoredDataCollection attribute</i>), 80	<i>dependencies()</i> (<i>propertyestimator.protocols.gradients.GradientReducedPotentials property</i>), 201
data (<i>propertyestimator.workflow.schemas.WorkflowDataCollectionToStore attribute</i>), 90	<i>dependencies()</i> (<i>propertyestimator.protocols.groups.ConditionalGroup property</i>), 215
decorrelate_statistics() (<i>propertyestimator.protocols.utils.BaseReweightingProtocols property</i>), 248	<i>dependencies()</i> (<i>propertyestimator.protocols.groups.ProtocolGroup property</i>), 210
decorrelate_trajectory() (<i>propertyestimator.protocols.utils.BaseReweightingProtocols property</i>), 248	<i>dependencies()</i> (<i>propertyestimator.protocols.miscellaneous.AddValues property</i>), 226
Density (class in <i>propertyestimator.properties</i>), 33	<i>dependencies()</i> (<i>propertyestimator.protocols.miscellaneous.DivideValue property</i>), 238
dependants_graph() (<i>propertyestimator.protocols.groups.ConditionalGroup property</i>), 215	<i>dependencies()</i> (<i>propertyestimator.protocols.miscellaneous.FilterSubstanceByRole property</i>), 241
dependants_graph() (<i>propertyestimator.protocols.groups.ProtocolGroup property</i>), 208	<i>dependencies()</i> (<i>propertyestimator.protocols.miscellaneous.MultiplyValue property</i>), 234
dependencies() (<i>propertyestimator.protocols.analysis.AveragePropertyProtocol property</i>), 152	<i>dependencies()</i> (<i>propertyestimator.protocols.miscellaneous.SubtractValues property</i>), 230
dependencies() (<i>propertyestimator.protocols.analysis.AverageTrajectoryProperty property</i>), 156	<i>dependencies()</i> (<i>propertyestimator.protocols.miscellaneous.WeightByMoleFraction property</i>), 245
dependencies() (<i>propertyestimator.protocols.analysis.ExtractAverageStatistic property</i>), 160	<i>dependencies()</i> (<i>propertyestimator.protocols.reweighting.BaseMBARProtocol property</i>), 190
dependencies() (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedData property</i>), 165	<i>dependencies()</i> (<i>propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM property</i>), 185
dependencies() (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData property</i>), 172	<i>dependencies()</i> (<i>propertyestimator.protocols.reweighting.ConcatenateStatistics property</i>), 180
dependencies() (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData property</i>), 168	<i>dependencies()</i> (<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories property</i>), 177
dependencies() (<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol property</i>), 102	<i>dependencies()</i> (<i>propertyestimator.protocols.reweighting.ReweightStatistics property</i>), 194
dependencies() (<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates property</i>), 111	<i>dependencies()</i> (<i>propertyestimator.protocols.simulation.RunEnergyMinimisation property</i>), 124
dependencies() (<i>propertyestimator.protocols.coordinates.SolvateExistingStructure property</i>), 106	<i>dependencies()</i> (<i>propertyestimator.protocols.simulation.RunOpenMMSimulation property</i>), 130
dependencies() (<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem property</i>), 115	

dependencies ()	(<i>propertyestimator.protocols.storage.UnpackStoredDataCollection</i> property), 218	effective_samples	(<i>propertyestimator.protocols.reweighting.ReweightStatistics</i> attribute), 188
dependencies ()	(<i>propertyestimator.protocols.storage.UnpackStoredSimulationData</i> property), 222	electrostatic_lambdas_1	(<i>propertyestimator.protocols.yank.SolvationYankProtocol</i> attribute), 145
dependencies ()	(<i>propertyestimator.protocols.yank.BaseYankProtocol</i> property), 135	electrostatic_lambdas_2	(<i>propertyestimator.protocols.yank.SolvationYankProtocol</i> attribute), 145
dependencies ()	(<i>propertyestimator.protocols.yank.LigandReceptorYankProtocol</i> property), 140	enable_pbc	(<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> attribute), 199
dependencies ()	(<i>propertyestimator.protocols.yank.SolvationYankProtocol</i> property), 147	enable_pbc	(<i>propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM</i> attribute), 183
dependencies ()	(<i>propertyestimator.workflow.protocols.BaseProtocol</i> property), 92	enable_pbc	(<i>propertyestimator.protocols.simulation.RunEnergyMinimisation</i> attribute), 123
DielectricConstant	(class in <i>propertyestimator.properties</i>), 37	enable_pbc	(<i>propertyestimator.protocols.simulation.RunOpenMMSimulation</i> attribute), 128
DivideValue	(class in <i>propertyestimator.protocols.miscellaneous</i>), 235	energy_minimisation ()	(<i>propertyestimator.protocols.utils.BaseSimulationProtocols</i> property), 250
divisor	(<i>propertyestimator.protocols.analysis.ExtractAverageStatistic</i> attribute), 159	ensemble	(<i>propertyestimator.protocols.simulation.RunOpenMMSimulation</i> attribute), 128
divisor	(<i>propertyestimator.protocols.miscellaneous.DivideValue</i> attribute), 236	EnthalpyOfMixing	(class in <i>propertyestimator.properties</i>), 40
docked_complex_coordinate_path	(<i>propertyestimator.protocols.coordinates.BuildDockedCoordinate</i> attribute), 110	EnthalpyOfVaporization	(class in <i>propertyestimator.properties</i>), 42
docked_ligand_coordinate_path	(<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates</i> attribute), 110	EnthalpyWorkflow	(<i>propertyestimator.properties.EnthalpyOfMixing</i> attribute), 40
doi	(<i>propertyestimator.properties.MeasurementSource</i> attribute), 31	equilibration_index	(<i>propertyestimator.protocols.analysis.AveragePropertyProtocol</i> attribute), 150
download_url ()	(<i>propertyestimator.forcefield.LigParGenForceFieldSource</i> property), 24	equilibration_index	(<i>propertyestimator.protocols.analysis.AverageTrajectoryProperty</i> attribute), 156
E		equilibration_index	(<i>propertyestimator.protocols.analysis.ExtractAverageStatistic</i> attribute), 161
effective_sample_indices	(<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> attribute), 199	equilibration_index	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedData</i> attribute), 163
effective_sample_indices	(<i>propertyestimator.protocols.reweighting.BaseMBARProtocol</i> attribute), 188	equilibration_index	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData</i> attribute), 173
effective_sample_indices	(<i>propertyestimator.protocols.reweighting.ReweightStatistics</i> attribute), 194	equilibration_index	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData</i> attribute), 169
effective_samples	(<i>propertyestimator.protocols.reweighting.BaseMBARProtocol</i>	equilibration_simulation ()	(<i>propertyesti-</i>

<i>mator.protocols.util.BaseSimulationProtocols</i> <i>property</i>), 250	<i>method</i>), 204
<i>estimated_free_energy</i> (<i>propertyestimator.protocols.yank.BaseYankProtocol</i> attribute), 134	<i>execute ()</i> (<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> method), 200
<i>estimated_free_energy</i> (<i>propertyestimator.protocols.yank.LigandReceptorYankProtocol</i> attribute), 140	<i>execute ()</i> (<i>propertyestimator.protocols.groups.ConditionalGroup</i> method), 212
<i>estimated_free_energy</i> (<i>propertyestimator.protocols.yank.SolvationYankProtocol</i> attribute), 147	<i>execute ()</i> (<i>propertyestimator.protocols.groups.ProtocolGroup</i> method), 209
<i>estimated_properties</i> (<i>propertyestimator.client.PropertyEstimatorResult</i> attribute), 17	<i>execute ()</i> (<i>propertyestimator.protocols.miscellaneous.AddValues</i> method), 225
<i>exceptions</i> (<i>propertyestimator.client.PropertyEstimatorResult</i> attribute), 17	<i>execute ()</i> (<i>propertyestimator.protocols.miscellaneous.DivideValue</i> method), 236
<i>ExcessMolarVolume</i> (class in <i>propertyestimator.properties</i>), 35	<i>execute ()</i> (<i>propertyestimator.protocols.miscellaneous.FilterSubstanceByRole</i> method), 240
<i>execute ()</i> (<i>propertyestimator.protocols.analysis.AveragePropertyProtocol</i> method), 151	<i>execute ()</i> (<i>propertyestimator.protocols.miscellaneous.MultiplyValue</i> method), 233
<i>execute ()</i> (<i>propertyestimator.protocols.analysis.AverageTrajectoryProperty</i> method), 155	<i>execute ()</i> (<i>propertyestimator.protocols.miscellaneous.SubtractValues</i> method), 229
<i>execute ()</i> (<i>propertyestimator.protocols.analysis.ExtractAverageStatistic</i> method), 159	<i>execute ()</i> (<i>propertyestimator.protocols.miscellaneous.WeightByMoleFraction</i> method), 244
<i>execute ()</i> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedData</i> method), 163	<i>execute ()</i> (<i>propertyestimator.protocols.reweighting.BaseMBARProtocol</i> method), 189
<i>execute ()</i> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData</i> method), 171	<i>execute ()</i> (<i>propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM</i> method), 184
<i>execute ()</i> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData</i> method), 167	<i>execute ()</i> (<i>propertyestimator.protocols.reweighting.ConcatenateStatistics</i> method), 179
<i>execute ()</i> (<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol</i> method), 101	<i>execute ()</i> (<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories</i> method), 176
<i>execute ()</i> (<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates</i> method), 112	<i>execute ()</i> (<i>propertyestimator.protocols.reweighting.ReweightStatistics</i> method), 193
<i>execute ()</i> (<i>propertyestimator.protocols.coordinates.SolvateExistingStructure</i> method), 104	<i>execute ()</i> (<i>propertyestimator.protocols.simulation.RunEnergyMinimisation</i> method), 123
<i>execute ()</i> (<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem</i> method), 114	<i>execute ()</i> (<i>propertyestimator.protocols.simulation.RunOpenMMSimulation</i> method), 129
<i>execute ()</i> (<i>propertyestimator.protocols.forcefield.BuildTLeapSystem</i> method), 118	<i>execute ()</i> (<i>propertyestimator.protocols.storage.UnpackStoredDataCollection</i> method), 216
<i>execute ()</i> (<i>propertyestimator.protocols.gradients.CentralDifferenceGradient</i>	<i>execute ()</i> (<i>propertyestimator.protocols.storage.UnpackStoredSimulationData</i>

- method*), 221
- `execute()` (*propertyestimator.protocols.yank.BaseYankProtocol method*), 134
- `execute()` (*propertyestimator.protocols.yank.LigandReceptorYankProtocol method*), 139
- `execute()` (*propertyestimator.protocols.yank.SolvationYankProtocol method*), 145
- `execute()` (*propertyestimator.workflow.protocols.BaseProtocol method*), 92
- `execution_order()` (*propertyestimator.protocols.groups.ConditionalGroup property*), 215
- `execution_order()` (*propertyestimator.protocols.groups.ProtocolGroup property*), 208
- `expected_components` (*propertyestimator.protocols.miscellaneous.FilterSubstanceByRole attribute*), 240
- `extract_uncorrelated_statistics()` (*propertyestimator.protocols.utils.BaseSimulationProtocols property*), 250
- `extract_uncorrelated_trajectory()` (*propertyestimator.protocols.utils.BaseSimulationProtocols property*), 250
- `ExtractAverageStatistic` (*class in propertyestimator.protocols.analysis*), 158
- `ExtractUncorrelatedData` (*class in propertyestimator.protocols.analysis*), 162
- `ExtractUncorrelatedStatisticsData` (*class in propertyestimator.protocols.analysis*), 170
- `ExtractUncorrelatedTrajectoryData` (*class in propertyestimator.protocols.analysis*), 166
- ## F
- `fidelity` (*propertyestimator.properties.CalculationSource attribute*), 32
- `filter_by_components()` (*propertyestimator.datasets.PhysicalPropertyDataSet method*), 55
- `filter_by_components()` (*propertyestimator.datasets.ThermoMLDataSet method*), 59
- `filter_by_elements()` (*propertyestimator.datasets.PhysicalPropertyDataSet method*), 56
- `filter_by_elements()` (*propertyestimator.datasets.ThermoMLDataSet method*), 59
- `filter_by_function()` (*propertyestimator.datasets.PhysicalPropertyDataSet method*), 54
- `filter_by_function()` (*propertyestimator.datasets.ThermoMLDataSet method*), 59
- `filter_by_phases()` (*propertyestimator.datasets.PhysicalPropertyDataSet method*), 54
- `filter_by_phases()` (*propertyestimator.datasets.ThermoMLDataSet method*), 59
- `filter_by_pressure()` (*propertyestimator.datasets.PhysicalPropertyDataSet method*), 55
- `filter_by_pressure()` (*propertyestimator.datasets.ThermoMLDataSet method*), 59
- `filter_by_property_types()` (*propertyestimator.datasets.PhysicalPropertyDataSet method*), 54
- `filter_by_property_types()` (*propertyestimator.datasets.ThermoMLDataSet method*), 60
- `filter_by_smiles()` (*propertyestimator.datasets.PhysicalPropertyDataSet method*), 56
- `filter_by_smiles()` (*propertyestimator.datasets.ThermoMLDataSet method*), 60
- `filter_by_temperature()` (*propertyestimator.datasets.PhysicalPropertyDataSet method*), 55
- `filter_by_temperature()` (*propertyestimator.datasets.ThermoMLDataSet method*), 60
- `filter_by_uncertainties()` (*propertyestimator.datasets.PhysicalPropertyDataSet method*), 56
- `filter_by_uncertainties()` (*propertyestimator.datasets.ThermoMLDataSet method*), 61
- `filtered_substance` (*propertyestimator.protocols.miscellaneous.FilterSubstanceByRole attribute*), 240
- `FilterSubstanceByRole` (*class in propertyestimator.protocols.miscellaneous*), 239
- `force_field_id` (*propertyestimator.storage.dataclasses.BaseStoredData attribute*), 77
- `force_field_path` (*propertyestimator.protocols.forcefield.BuildSmirnoffSystem attribute*), 115
- `force_field_path` (*propertyestimator*), 115

<i>tor.protocols.forcefield.BuildTLeapSystem</i> attribute), 120	61	<i>full_path()</i> (<i>propertyestimator.workflow.utils.ProtocolPath</i> property),
<i>force_field_path</i> (<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> attribute), 198	96	<i>full_substance</i> (<i>propertyestimator.protocols.miscellaneous.WeightByMoleFraction</i> attribute), 244
<i>force_field_path</i> (<i>propertyestimator.protocols.storage.UnpackStoredSimulationData</i> attribute), 221		
<i>force_field_path</i> (<i>propertyestimator.protocols.yank.LigandReceptorYankProtocol</i> attribute), 139	G	<i>generate_base_reweighting_protocols()</i> (in module <i>propertyestimator.protocols.utils</i>), 250
<i>force_field_source</i> (<i>propertyestimator.client.PropertyEstimatorSubmission</i> attribute), 15		<i>generate_base_simulation_protocols()</i> (in module <i>propertyestimator.protocols.utils</i>), 251
<i>ForceFieldSource</i> (class in <i>propertyestimator.forcefield</i>), 19		<i>generate_default_metadata()</i> (<i>propertyestimator.workflow.Workflow</i> static method), 81
<i>forward_observable_value</i> (<i>propertyestimator.protocols.gradients.CentralDifferenceGradient</i> attribute), 204		<i>generate_gradient_protocol_group()</i> (in module <i>propertyestimator.protocols.utils</i>), 252
<i>forward_parameter_value</i> (<i>propertyestimator.protocols.gradients.CentralDifferenceGradient</i> attribute), 204		<i>get_amounts()</i> (<i>propertyestimator.substances.Substance</i> method), 50
<i>forward_parameter_value</i> (<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> attribute), 200		<i>get_class_attribute()</i> (<i>propertyestimator.protocols.analysis.AveragePropertyProtocol</i> method), 152
<i>forward_potentials_path</i> (<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> attribute), 200		<i>get_class_attribute()</i> (<i>propertyestimator.protocols.analysis.AverageTrajectoryProperty</i> method), 156
<i>frame_counts</i> (<i>propertyestimator.protocols.reweighting.ReweightStatistics</i> attribute), 193		<i>get_class_attribute()</i> (<i>propertyestimator.protocols.analysis.ExtractAverageStatistic</i> method), 161
<i>from_components()</i> (<i>propertyestimator.substances.Substance</i> class method), 50		<i>get_class_attribute()</i> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedData</i> method), 165
<i>from_doi()</i> (<i>propertyestimator.datasets.ThermoMLDataSet</i> class method), 58		<i>get_class_attribute()</i> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData</i> method), 173
<i>from_file()</i> (<i>propertyestimator.datasets.ThermoMLDataSet</i> class method), 58		<i>get_class_attribute()</i> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData</i> method), 169
<i>from_json()</i> (<i>propertyestimator.client.PropertyEstimatorClient.Request</i> class method), 13		<i>get_class_attribute()</i> (<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol</i> method), 102
<i>from_object()</i> (<i>propertyestimator.forcefield.SmirnoffForceFieldSource</i> class method), 20		<i>get_class_attribute()</i> (<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates</i> method), 111
<i>from_path()</i> (<i>propertyestimator.forcefield.SmirnoffForceFieldSource</i> class method), 20		<i>get_class_attribute()</i> (<i>propertyestimator.protocols.coordinates.SolvateExistingStructure</i> method), 106
<i>from_url()</i> (<i>propertyestimator.datasets.ThermoMLDataSet</i> class method), 58		<i>get_class_attribute()</i> (<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem</i> method), 116
<i>from_xml()</i> (<i>propertyestimator.datasets.ThermoMLDataSet</i> class method),		<i>get_class_attribute()</i> (<i>propertyestimator.protocols.forcefield.BuildTLeapSystem</i> method), 120

get_class_attribute() (propertyestimator.protocols.gradients.CentralDifferenceGradient method), 205	get_class_attribute() (propertyestimator.protocols.storage.UnpackStoredSimulationData method), 222
get_class_attribute() (propertyestimator.protocols.gradients.GradientReducedPotentials method), 201	get_class_attribute() (propertyestimator.protocols.yank.BaseYankProtocol method), 135
get_class_attribute() (propertyestimator.protocols.groups.ConditionalGroup method), 213	get_class_attribute() (propertyestimator.protocols.yank.LigandReceptorYankProtocol method), 140
get_class_attribute() (propertyestimator.protocols.groups.ProtocolGroup method), 209	get_class_attribute() (propertyestimator.protocols.yank.SolvationYankProtocol method), 147
get_class_attribute() (propertyestimator.protocols.miscellaneous.AddValues method), 226	get_class_attribute() (propertyestimator.workflow.protocols.BaseProtocol method), 93
get_class_attribute() (propertyestimator.protocols.miscellaneous.DivideValue method), 238	get_default_reweighting_workflow_schema() (propertyestimator.properties.Density static method), 34
get_class_attribute() (propertyestimator.protocols.miscellaneous.FilterSubstanceByRole method), 241	get_default_reweighting_workflow_schema() (propertyestimator.properties.DielectricConstant static method), 39
get_class_attribute() (propertyestimator.protocols.miscellaneous.MultiplyValue method), 234	get_default_reweighting_workflow_schema() (propertyestimator.properties.EnthalpyOfMixing static method), 41
get_class_attribute() (propertyestimator.protocols.miscellaneous.SubtractValues method), 230	get_default_reweighting_workflow_schema() (propertyestimator.properties.EnthalpyOfVaporization static method), 43
get_class_attribute() (propertyestimator.protocols.miscellaneous.WeightByMoleFraction method), 245	get_default_reweighting_workflow_schema() (propertyestimator.properties.ExcessMolarVolume static method), 36
get_class_attribute() (propertyestimator.protocols.reweighting.BaseMBARProtocol method), 190	get_default_simulation_workflow_schema() (propertyestimator.properties.Density static method), 34
get_class_attribute() (propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM method), 185	get_default_simulation_workflow_schema() (propertyestimator.properties.DielectricConstant static method), 38
get_class_attribute() (propertyestimator.protocols.reweighting.ConcatenateStatistics method), 181	get_default_simulation_workflow_schema() (propertyestimator.properties.EnthalpyOfMixing static method), 41
get_class_attribute() (propertyestimator.protocols.reweighting.ConcatenateTrajectories method), 177	get_default_simulation_workflow_schema() (propertyestimator.properties.EnthalpyOfVaporization static method), 43
get_class_attribute() (propertyestimator.protocols.reweighting.ReweightStatistics method), 195	get_default_simulation_workflow_schema() (propertyestimator.properties.ExcessMolarVolume static method), 36
get_class_attribute() (propertyestimator.protocols.simulation.RunEnergyMinimisation method), 124	get_default_simulation_workflow_schema() (propertyestimator.properties.ExcessMolarVolume static method), 36
get_class_attribute() (propertyestimator.protocols.simulation.RunOpenMMSimulation method), 130	get_default_simulation_workflow_schema() (propertyestimator.properties.ExcessMolarVolume static method), 36
get_class_attribute() (propertyestimator.protocols.storage.UnpackStoredDataCollection method), 218	get_default_simulation_workflow_schema() (propertyestimator.properties.ExcessMolarVolume static method), 36

(*propertyestimator.properties.HostGuestBindingAffinity static method*), 45

`get_default_workflow_schema()` (*propertyestimator.properties.Density static method*), 34

`get_default_workflow_schema()` (*propertyestimator.properties.DielectricConstant static method*), 38

`get_default_workflow_schema()` (*propertyestimator.properties.EnthalpyOfMixing static method*), 40

`get_default_workflow_schema()` (*propertyestimator.properties.EnthalpyOfVaporization static method*), 43

`get_default_workflow_schema()` (*propertyestimator.properties.ExcessMolarVolume static method*), 36

`get_default_workflow_schema()` (*propertyestimator.properties.HostGuestBindingAffinity static method*), 45

`get_default_workflow_schema()` (*propertyestimator.properties.PhysicalProperty static method*), 29

`get_molecules_per_component()` (*propertyestimator.substances.Substance method*), 50

`get_value()` (*propertyestimator.protocols.analysis.AveragePropertyProtocol method*), 152

`get_value()` (*propertyestimator.protocols.analysis.AverageTrajectoryProperty method*), 156

`get_value()` (*propertyestimator.protocols.analysis.ExtractAverageStatistic method*), 161

`get_value()` (*propertyestimator.protocols.analysis.ExtractUncorrelatedData method*), 165

`get_value()` (*propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData method*), 173

`get_value()` (*propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData method*), 169

`get_value()` (*propertyestimator.protocols.coordinates.BuildCoordinatesPackmol method*), 102

`get_value()` (*propertyestimator.protocols.coordinates.BuildDockedCoordinates method*), 111

`get_value()` (*propertyestimator.protocols.coordinates.SolvateExistingStructure method*), 106

`get_value()` (*propertyestimator.protocols.forcefield.BuildSmirnoffSystem method*), 116

`get_value()` (*propertyestimator.protocols.forcefield.BuildTLeapSystem method*), 120

`get_value()` (*propertyestimator.protocols.gradients.CentralDifferenceGradient method*), 206

`get_value()` (*propertyestimator.protocols.gradients.GradientReducedPotentials method*), 201

`get_value()` (*propertyestimator.protocols.groups.ConditionalGroup method*), 213

`get_value()` (*propertyestimator.protocols.groups.ProtocolGroup method*), 209

`get_value()` (*propertyestimator.protocols.miscellaneous.AddValues method*), 226

`get_value()` (*propertyestimator.protocols.miscellaneous.DivideValue method*), 238

`get_value()` (*propertyestimator.protocols.miscellaneous.FilterSubstanceByRole method*), 242

`get_value()` (*propertyestimator.protocols.miscellaneous.MultiplyValue method*), 234

`get_value()` (*propertyestimator.protocols.miscellaneous.SubtractValues method*), 230

`get_value()` (*propertyestimator.protocols.miscellaneous.WeightByMoleFraction method*), 246

`get_value()` (*propertyestimator.protocols.reweighting.BaseMBARProtocol method*), 190

`get_value()` (*propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM method*), 185

`get_value()` (*propertyestimator.protocols.reweighting.ConcatenateStatistics method*), 181

`get_value()` (*propertyestimator.protocols.reweighting.ConcatenateTrajectories method*), 177

`get_value()` (*propertyestimator.protocols.reweighting.ReweightStatistics method*), 195

`get_value()` (*propertyestimator.protocols.simulation.RunEnergyMinimisation method*), 125

`get_value()` (*propertyestimator.protocols.simulation.RunOpenMMSimulation method*), 125

<i>method</i>), 130		<i>method</i>), 206	
get_value ()	(propertyestimator.protocols.storage.UnpackStoredDataCollection method), 218	get_value_references ()	(propertyestimator.protocols.gradients.GradientReducedPotentials method), 201
get_value ()	(propertyestimator.protocols.storage.UnpackStoredSimulationData method), 222	get_value_references ()	(propertyestimator.protocols.groups.ConditionalGroup method), 214
get_value ()	(propertyestimator.protocols.yank.BaseYankProtocol method), 135	get_value_references ()	(propertyestimator.protocols.groups.ProtocolGroup method), 210
get_value ()	(propertyestimator.protocols.yank.LigandReceptorYankProtocol method), 141	get_value_references ()	(propertyestimator.protocols.miscellaneous.AddValues method), 226
get_value ()	(propertyestimator.protocols.yank.SolvationYankProtocol method), 147	get_value_references ()	(propertyestimator.protocols.miscellaneous.DivideValue method), 238
get_value ()	(propertyestimator.workflow.protocols.BaseProtocol method), 93	get_value_references ()	(propertyestimator.protocols.miscellaneous.FilterSubstanceByRole method), 242
get_value_references ()	(propertyestimator.protocols.analysis.AveragePropertyProtocol method), 152	get_value_references ()	(propertyestimator.protocols.miscellaneous.MultiplyValue method), 234
get_value_references ()	(propertyestimator.protocols.analysis.AverageTrajectoryProperty method), 156	get_value_references ()	(propertyestimator.protocols.miscellaneous.SubtractValues method), 230
get_value_references ()	(propertyestimator.protocols.analysis.ExtractAverageStatistic method), 161	get_value_references ()	(propertyestimator.protocols.miscellaneous.WeightByMoleFraction method), 246
get_value_references ()	(propertyestimator.protocols.analysis.ExtractUncorrelatedData method), 165	get_value_references ()	(propertyestimator.protocols.reweighting.BaseMBARProtocol method), 190
get_value_references ()	(propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData method), 173	get_value_references ()	(propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM method), 185
get_value_references ()	(propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData method), 169	get_value_references ()	(propertyestimator.protocols.reweighting.ConcatenateStatistics method), 181
get_value_references ()	(propertyestimator.protocols.coordinates.BuildCoordinatesPackmol method), 102	get_value_references ()	(propertyestimator.protocols.reweighting.ConcatenateTrajectories method), 177
get_value_references ()	(propertyestimator.protocols.coordinates.BuildDockedCoordinates method), 111	get_value_references ()	(propertyestimator.protocols.reweighting.ReweightStatistics method), 195
get_value_references ()	(propertyestimator.protocols.coordinates.SolvateExistingStructure method), 106	get_value_references ()	(propertyestimator.protocols.simulation.RunEnergyMinimisation method), 125
get_value_references ()	(propertyestimator.protocols.forcefield.BuildSmirnoffSystem method), 116	get_value_references ()	(propertyestimator.protocols.simulation.RunOpenMMSimulation method), 130
get_value_references ()	(propertyestimator.protocols.forcefield.BuildTLeapSystem method), 120	get_value_references ()	(propertyestimator.protocols.storage.UnpackStoredDataCollection method), 218
get_value_references ()	(propertyestimator.protocols.gradients.CentralDifferenceGradient	get_value_references ()	(propertyestimator.protocols.storage.UnpackStoredSimulationData

<i>method</i>), 222			
<code>get_value_references()</code>	(<i>propertyestimator.protocols.yank.BaseYankProtocol method</i>), 135	<code>id()</code>	(<i>propertyestimator.protocols.analysis.ExtractAverageStatistic property</i>), 161
<code>get_value_references()</code>	(<i>propertyestimator.protocols.yank.LigandReceptorYankProtocol method</i>), 141	<code>id()</code>	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedData property</i>), 165
<code>get_value_references()</code>	(<i>propertyestimator.protocols.yank.SolvationYankProtocol method</i>), 147	<code>id()</code>	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData property</i>), 173
<code>get_value_references()</code>	(<i>propertyestimator.workflow.protocols.BaseProtocol method</i>), 93	<code>id()</code>	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData property</i>), 169
<code>gpu_device_indices()</code>	(<i>propertyestimator.backends.ComputeResources property</i>), 67	<code>id()</code>	(<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol property</i>), 102
<code>gpu_device_indices()</code>	(<i>propertyestimator.backends.QueueWorkerResources property</i>), 69	<code>id()</code>	(<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates property</i>), 111
<code>gradient</code>	(<i>propertyestimator.protocols.gradients.CentralDifferenceGradient attribute</i>), 204	<code>id()</code>	(<i>propertyestimator.protocols.coordinates.SolvateExistingStructure property</i>), 106
<code>GradientReducedPotentials</code>	(<i>class in propertyestimator.protocols.gradients</i>), 197	<code>id()</code>	(<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem property</i>), 116
H		<code>id()</code>	(<i>propertyestimator.protocols.forcefield.BuildTLeapSystem property</i>), 121
<code>handle_stream()</code>	(<i>propertyestimator.server.PropertyEstimatorServer method</i>), 27	<code>id()</code>	(<i>propertyestimator.protocols.gradients.CentralDifferenceGradient property</i>), 206
<code>has_force_field()</code>	(<i>propertyestimator.storage.LocalFileStorage method</i>), 76	<code>id()</code>	(<i>propertyestimator.protocols.gradients.GradientReducedPotentials property</i>), 202
<code>has_force_field()</code>	(<i>propertyestimator.storage.PropertyEstimatorStorage method</i>), 73	<code>id()</code>	(<i>propertyestimator.protocols.groups.ConditionalGroup property</i>), 215
<code>high_precision</code>	(<i>propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM attribute</i>), 184	<code>id()</code>	(<i>propertyestimator.protocols.groups.ProtocolGroup property</i>), 211
<code>high_precision</code>	(<i>propertyestimator.protocols.simulation.RunOpenMMSimulation attribute</i>), 129	<code>id()</code>	(<i>propertyestimator.protocols.miscellaneous.AddValues property</i>), 227
<code>HostGuestBindingAffinity</code>	(<i>class in propertyestimator.properties</i>), 44	<code>id()</code>	(<i>propertyestimator.protocols.miscellaneous.DivideValue property</i>), 238
I		<code>id()</code>	(<i>propertyestimator.protocols.miscellaneous.FilterSubstanceByRole property</i>), 242
<code>id</code>	(<i>propertyestimator.client.PropertyEstimatorResult attribute</i>), 16	<code>id()</code>	(<i>propertyestimator.protocols.miscellaneous.MultiplyValue property</i>), 234
<code>id()</code>	(<i>propertyestimator.client.PropertyEstimatorClient.Request property</i>), 13	<code>id()</code>	(<i>propertyestimator.protocols.miscellaneous.SubtractValues property</i>), 157
<code>id()</code>	(<i>propertyestimator.protocols.analysis.AveragePropertyProtocol property</i>), 153	<code>id()</code>	
<code>id()</code>	(<i>propertyestimator.protocols.analysis.AverageTrajectoryProperty property</i>), 157	<code>id()</code>	

id()	<i>property</i>), 231	identifier()	(<i>propertyestimator.substances.Substance.MoleFraction property</i>), 49
id()	(<i>propertyestimator.protocols.miscellaneous.WeightByMoleFraction property</i>), 246	index()	(<i>propertyestimator.protocols.utils.BaseReweightingProtocols method</i>), 248
id()	(<i>propertyestimator.protocols.reweighting.BaseMBARProtocol property</i>), 190	index()	(<i>propertyestimator.protocols.utils.BaseSimulationProtocols method</i>), 250
id()	(<i>propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM property</i>), 186	InequalityMergeBehaviour	(class in <i>propertyestimator.workflow.decorators</i>), 98
id()	(<i>propertyestimator.protocols.reweighting.ConcatenateStatistics property</i>), 181	input_coordinate_file	(<i>propertyestimator.protocols.analysis.AverageTrajectoryProperty attribute</i>), 155
id()	(<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories property</i>), 177	input_coordinate_file	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData attribute</i>), 167
id()	(<i>propertyestimator.protocols.reweighting.ReweightStatistics property</i>), 195	input_coordinate_file	(<i>propertyestimator.protocols.simulation.RunEnergyMinimisation attribute</i>), 123
id()	(<i>propertyestimator.protocols.simulation.RunEnergyMinimisation property</i>), 125	input_coordinate_file	(<i>propertyestimator.protocols.simulation.RunOpenMMSimulation attribute</i>), 128
id()	(<i>propertyestimator.protocols.simulation.RunOpenMMSimulation property</i>), 131	input_coordinate_paths	(<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories attribute</i>), 175
id()	(<i>propertyestimator.protocols.storage.UnpackStoredDataCollection property</i>), 218	input_data_path	(<i>propertyestimator.protocols.storage.UnpackStoredDataCollection attribute</i>), 216
id()	(<i>propertyestimator.protocols.storage.UnpackStoredSimulationData property</i>), 223	input_statistics_path	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData attribute</i>), 171
id()	(<i>propertyestimator.protocols.yank.BaseYankProtocol property</i>), 136	input_statistics_paths	(<i>propertyestimator.protocols.reweighting.ConcatenateStatistics attribute</i>), 179
id()	(<i>propertyestimator.protocols.yank.LigandReceptorYankProtocol property</i>), 141	input_substance	(<i>propertyestimator.protocols.miscellaneous.FilterSubstanceByRole attribute</i>), 240
id()	(<i>propertyestimator.protocols.yank.SolvationYankProtocol property</i>), 147	input_trajectory_path	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData attribute</i>), 167
id()	(<i>propertyestimator.workflow.protocols.BaseProtocol property</i>), 92	input_trajectory_paths	(<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories attribute</i>), 175
identifier()	(<i>propertyestimator.substances.Substance property</i>), 49	inverse_beta()	(<i>propertyestimator.thermodynamics.ThermodynamicState property</i>), 52
identifier()	(<i>propertyestimator.substances.Substance.Amount property</i>), 48	IWorkflowProperty	(class in <i>propertyestimator.workflow</i>), 84
identifier()	(<i>propertyestimator.substances.Substance.Component property</i>), 48		
identifier()	(<i>propertyestimator.substances.Substance.ExactAmount property</i>), 49	job_script()	(<i>propertyestimator.backends.DaskLSFBackend method</i>), 72

J

- `json()` (*propertyestimator.client.ConnectionOptions* method), 18
- `json()` (*propertyestimator.client.PropertyEstimatorClient.Request* method), 13
- `json()` (*propertyestimator.client.PropertyEstimatorOptions* method), 15
- `json()` (*propertyestimator.client.PropertyEstimatorResult* method), 17
- `json()` (*propertyestimator.client.PropertyEstimatorSubmission* method), 16
- `json()` (*propertyestimator.datasets.PhysicalPropertyDataSet* method), 57
- `json()` (*propertyestimator.datasets.ThermoMLDataSet* method), 61
- `json()` (*propertyestimator.forcefield.ForceFieldSource* method), 19
- `json()` (*propertyestimator.forcefield.LigParGenForceFieldSource* method), 24
- `json()` (*propertyestimator.forcefield.SmirnoffForceFieldSource* method), 21
- `json()` (*propertyestimator.forcefield.TLeapForceFieldSource* method), 22
- `json()` (*propertyestimator.properties.CalculationSource* method), 33
- `json()` (*propertyestimator.properties.Density* method), 34
- `json()` (*propertyestimator.properties.DielectricConstant* method), 39
- `json()` (*propertyestimator.properties.EnthalpyOfMixing* method), 41
- `json()` (*propertyestimator.properties.EnthalpyOfVaporization* method), 43
- `json()` (*propertyestimator.properties.ExcessMolarVolume* method), 37
- `json()` (*propertyestimator.properties.HostGuestBindingAffinity* method), 45
- `json()` (*propertyestimator.properties.MeasurementSource* method), 32
- `json()` (*propertyestimator.properties.PhysicalProperty* method), 30
- `json()` (*propertyestimator.properties.Source* method), 31
- `json()` (*propertyestimator.server.PropertyEstimatorServer.ServerEstimationRequest* method), 26
- `json()` (*propertyestimator.substances.Substance* method), 51
- `json()` (*propertyestimator.substances.Substance.Component* method), 48
- `json()` (*propertyestimator.thermodynamics.ThermodynamicState* method), 52
- `json()` (*propertyestimator.workflow.schemas.ProtocolGroupSchema* method), 86
- `json()` (*propertyestimator.workflow.schemas.ProtocolReplicator* method), 88
- `json()` (*propertyestimator.workflow.schemas.ProtocolSchema* method), 86
- `json()` (*propertyestimator.workflow.schemas.WorkflowSchema* method), 85
- ## K
- `kinetic_energies_path` (*propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM* attribute), 184
- ## L
- `label()` (*propertyestimator.substances.Substance.Component* property), 48
- `last_protocol()` (*propertyestimator.workflow.utils.ProtocolPath* property), 96
- `leap_source()` (*propertyestimator.forcefield.TLeapForceFieldSource* property), 22
- `ligand_residue_name` (*propertyestimator.protocols.coordinates.BuildDockedCoordinates* attribute), 110
- `ligand_residue_name` (*propertyestimator.protocols.yank.LigandReceptorYankProtocol* attribute), 138
- `ligand_substance` (*propertyestimator.protocols.coordinates.BuildDockedCoordinates* attribute), 109
- `LigandReceptorYankProtocol` (class in *propertyestimator.protocols.yank*), 136

LigandReceptorYankProtocol.RestraintType (class in <i>propertyestimator.protocols.yank</i>), 138	<i>merge()</i>	<i>method</i>), 165 (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData</i> <i>method</i>), 173
LigParGenForceFieldSource (class in <i>propertyestimator.forcefield</i>), 22	<i>merge()</i>	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData</i> <i>method</i>), 169
LigParGenForceFieldSource.ChargeModel (class in <i>propertyestimator.forcefield</i>), 23	<i>merge()</i>	(<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol</i> <i>method</i>), 102
listen() (propertyestimator.server.PropertyEstimatorServer <i>method</i>), 28	<i>merge()</i>	(<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates</i> <i>method</i>), 112
LocalFileStorage (class in <i>propertyestimator.storage</i>), 75	<i>merge()</i>	(<i>propertyestimator.protocols.coordinates.SolvateExistingStructure</i> <i>method</i>), 107
M	<i>merge()</i>	(<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem</i> <i>method</i>), 116
mass_density (propertyestimator.protocols.coordinates.BuildCoordinatesPackmol <i>attribute</i>), 100	<i>merge()</i>	(<i>propertyestimator.protocols.forcefield.BuildTLeapSystem</i> <i>method</i>), 121
mass_density (propertyestimator.protocols.coordinates.SolvateExistingStructure <i>attribute</i>), 107	<i>merge()</i>	(<i>propertyestimator.protocols.gradients.CentralDifferenceGradient</i> <i>method</i>), 206
max_iterations (propertyestimator.protocols.groups.ConditionalGroup <i>attribute</i>), 212	<i>merge()</i>	(<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> <i>method</i>), 202
max_iterations (propertyestimator.protocols.simulation.RunEnergyMinimisation <i>attribute</i>), 123	<i>merge()</i>	(<i>propertyestimator.protocols.groups.ConditionalGroup</i> <i>method</i>), 213
max_molecules (propertyestimator.protocols.coordinates.BuildCoordinatesPackmol <i>attribute</i>), 100	<i>merge()</i>	(<i>propertyestimator.protocols.groups.ProtocolGroup</i> <i>method</i>), 209
max_molecules (propertyestimator.protocols.coordinates.SolvateExistingStructure <i>attribute</i>), 107	<i>merge()</i>	(<i>propertyestimator.protocols.miscellaneous.AddValues</i> <i>method</i>), 227
mbar_protocol() (propertyestimator.protocols.utils.BaseReweightingProtocols <i>property</i>), 248	<i>merge()</i>	(<i>propertyestimator.protocols.miscellaneous.DivideValue</i> <i>method</i>), 238
MeasurementSource (class in <i>propertyestimator.properties</i>), 31	<i>merge()</i>	(<i>propertyestimator.protocols.miscellaneous.FilterSubstanceByRole</i> <i>method</i>), 242
<i>merge()</i> (propertyestimator.datasets.PhysicalPropertyDataSet <i>method</i>), 54	<i>merge()</i>	(<i>propertyestimator.protocols.miscellaneous.MultiplyValue</i> <i>method</i>), 234
<i>merge()</i> (propertyestimator.datasets.ThermoMLDataSet <i>method</i>), 61	<i>merge()</i>	(<i>propertyestimator.protocols.miscellaneous.SubtractValues</i> <i>method</i>), 231
<i>merge()</i> (propertyestimator.protocols.analysis.AveragePropertyProtocol <i>method</i>), 153	<i>merge()</i>	(<i>propertyestimator.protocols.miscellaneous.WeightByMoleFraction</i> <i>method</i>), 246
<i>merge()</i> (propertyestimator.protocols.analysis.AverageTrajectoryProperty <i>method</i>), 157	<i>merge()</i>	(<i>propertyestimator.protocols.reweighting.BaseMBARProtocol</i>
<i>merge()</i> (propertyestimator.protocols.analysis.ExtractAverageStatistic <i>method</i>), 161	<i>merge()</i>	
<i>merge()</i> (propertyestimator.protocols.analysis.ExtractUncorrelatedData	<i>merge()</i>	

method), 190

merge () (propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM method), 186

merge () (propertyestimator.protocols.reweighting.ConcatenateStatistics method), 181

merge () (propertyestimator.protocols.reweighting.ConcatenateTrajectories method), 178

merge () (propertyestimator.protocols.reweighting.ReweightStatistics method), 195

merge () (propertyestimator.protocols.simulation.RunEnergyMinimisation method), 125

merge () (propertyestimator.protocols.simulation.RunOpenMMSimulation method), 131

merge () (propertyestimator.protocols.storage.UnpackStoredDataCollection method), 218

merge () (propertyestimator.protocols.storage.UnpackStoredSimulationData method), 223

merge () (propertyestimator.protocols.yank.BaseYankProtocol method), 136

merge () (propertyestimator.protocols.yank.LigandReceptorYankProtocol method), 141

merge () (propertyestimator.protocols.yank.SolvationYankProtocol method), 148

merge () (propertyestimator.storage.dataclasses.BaseStoredData class method), 78

merge () (propertyestimator.storage.dataclasses.StoredDataCollection class method), 80

merge () (propertyestimator.storage.dataclasses.StoredSimulationData class method), 79

merge () (propertyestimator.workflow.protocols.BaseProtocol method), 93

MergeBehaviour (class in propertyestimator.workflow.decorators), 98

metadata () (propertyestimator.properties.Density property), 35

metadata () (propertyestimator.properties.DielectricConstant property), 39

metadata () (propertyestimator.properties.EnthalpyOfMixing property), 41

metadata () (propertyestimator.properties.EnthalpyOfVaporization property), 43

metadata () (propertyestimator.properties.ExcessMolarVolume property), 37

metadata () (propertyestimator.properties.HostGuestBindingAffinity property), 45

metadata () (propertyestimator.properties.PhysicalProperty property), 29

multi_component_property () (propertyestimator.properties.EnthalpyOfVaporization property), 43

multi_component_property () (propertyestimator.properties.HostGuestBindingAffinity property), 45

multiplier (propertyestimator.protocols.miscellaneous.MultiplyValue attribute), 232

MultiplyValue (class in propertyestimator.protocols.miscellaneous), 231

N

number_of_components () (propertyestimator.substances.Substance property), 50

number_of_equilibration_iterations (propertyestimator.protocols.yank.BaseYankProtocol attribute), 133

number_of_equilibration_iterations (propertyestimator.protocols.yank.LigandReceptorYankProtocol attribute), 141

number_of_equilibration_iterations (propertyestimator.protocols.yank.SolvationYankProtocol attribute), 148

number_of_gpus () (propertyestimator.backends.ComputeResources property), 67

number_of_gpus () (propertyestimator.backends.QueueWorkerResources property), 69

number_of_iterations (propertyestimator.protocols.yank.BaseYankProtocol attribute), 133

number_of_iterations (propertyestimator.protocols.yank.LigandReceptorYankProtocol attribute), 141

number_of_iterations (propertyestimator.properties.EnthalpyOfMixing property), 41

number_of_iterations (propertyestimator.properties.EnthalpyOfVaporization property), 43

number_of_iterations (propertyestimator.properties.ExcessMolarVolume property), 37

number_of_iterations (propertyestimator.properties.HostGuestBindingAffinity property), 45

number_of_iterations (propertyestimator.properties.PhysicalProperty property), 29

number_of_iterations (propertyestimator.properties.EnthalpyOfVaporization property), 43

number_of_iterations (propertyestimator.properties.HostGuestBindingAffinity property), 45

number_of_iterations (propertyestimator.properties.PhysicalProperty property), 29

multi_component_property () (propertyestimator.properties.EnthalpyOfVaporization property), 43

multi_component_property () (propertyestimator.properties.HostGuestBindingAffinity property), 45

multiplier (propertyestimator.protocols.miscellaneous.MultiplyValue attribute), 232

MultiplyValue (class in propertyestimator.protocols.miscellaneous), 231

<i>tor.protocols.yank.SolvationYankProtocol</i> attribute), 148	<i>tor.protocols.analysis.ExtractUncorrelatedStatisticsData</i> attribute), 171
number_of_ligand_conformers (<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates</i> attribute), 109	output_statistics_path (<i>propertyestimator.protocols.reweighting.ConcatenateStatistics</i> attribute), 179
number_of_properties() (<i>propertyestimator.datasets.PhysicalPropertyDataSet</i> property), 54	output_substance (<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol</i> attribute), 100
number_of_properties() (<i>propertyestimator.datasets.ThermoMLDataSet</i> property), 61	output_substance (<i>propertyestimator.protocols.coordinates.SolvateExistingStructure</i> attribute), 107
number_of_threads() (<i>propertyestimator.backends.ComputeResources</i> property), 67	output_trajectory_path (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData</i> attribute), 167
number_of_threads() (<i>propertyestimator.backends.QueueWorkerResources</i> property), 69	output_trajectory_path (<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories</i> attribute), 176
number_of_uncorrelated_samples (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedData</i> attribute), 163	P
number_of_uncorrelated_samples (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData</i> attribute), 173	parameter_key (<i>propertyestimator.protocols.gradients.CentralDifferenceGradient</i> attribute), 204
number_of_uncorrelated_samples (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData</i> attribute), 169	parameter_key (<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> attribute), 199
O	ParameterGradient (class in <i>propertyestimator.properties</i>), 25
options (<i>propertyestimator.client.PropertyEstimatorSubmission</i> attribute), 15	ParameterGradientKey (class in <i>propertyestimator.properties</i>), 24
output_coordinate_file (<i>propertyestimator.protocols.simulation.RunEnergyMinimisation</i> attribute), 123	parse_json() (<i>propertyestimator.client.ConnectionOptions</i> class method), 18
output_coordinate_file (<i>propertyestimator.protocols.simulation.RunOpenMMSimulation</i> attribute), 129	parse_json() (<i>propertyestimator.client.PropertyEstimatorOptions</i> class method), 15
output_coordinate_path (<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories</i> attribute), 175	parse_json() (<i>propertyestimator.client.PropertyEstimatorResult</i> class method), 17
output_frequency (<i>propertyestimator.protocols.simulation.RunOpenMMSimulation</i> attribute), 128	parse_json() (<i>propertyestimator.client.PropertyEstimatorSubmission</i> class method), 16
output_number_of_molecules (<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol</i> attribute), 100	parse_json() (<i>propertyestimator.datasets.PhysicalPropertyDataSet</i> class method), 57
output_number_of_molecules (<i>propertyestimator.protocols.coordinates.SolvateExistingStructure</i> attribute), 107	parse_json() (<i>propertyestimator.datasets.ThermoMLDataSet</i> class method), 61
output_statistics_path (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedData</i> attribute), 163	parse_json() (<i>propertyestimator.forcefield.ForceFieldSource</i> class method), 19
output_statistics_path (<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories</i> attribute), 176	parse_json() (<i>propertyestimator.forcefield.LigParGenForceFieldSource</i> class method), 24
output_statistics_path (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData</i> attribute), 171	parse_json() (<i>propertyestimator</i> class method), 24

tor.forcefield.SmirnoffForceFieldSource class method), 21
 parse_json() (*propertyestimator.forcefield.TLeapForceFieldSource* class method), 22
 parse_json() (*propertyestimator.properties.CalculationSource* class method), 33
 parse_json() (*propertyestimator.properties.Density* class method), 35
 parse_json() (*propertyestimator.properties.DielectricConstant* class method), 39
 parse_json() (*propertyestimator.properties.EnthalpyOfMixing* class method), 41
 parse_json() (*propertyestimator.properties.EnthalpyOfVaporization* class method), 43
 parse_json() (*propertyestimator.properties.ExcessMolarVolume* class method), 37
 parse_json() (*propertyestimator.properties.HostGuestBindingAffinity* class method), 46
 parse_json() (*propertyestimator.properties.MeasurementSource* class method), 32
 parse_json() (*propertyestimator.properties.PhysicalProperty* class method), 30
 parse_json() (*propertyestimator.properties.Source* class method), 31
 parse_json() (*propertyestimator.server.PropertyEstimatorServer.ServerEstimationRequestor.thermodynamics.ThermodynamicState* class method), 27
 parse_json() (*propertyestimator.substances.Substance* class method), 51
 parse_json() (*propertyestimator.substances.Substance.Component* class method), 48
 parse_json() (*propertyestimator.thermodynamics.ThermodynamicState* class method), 52
 parse_json() (*propertyestimator.workflow.schemas.ProtocolGroupSchema* class method), 86
 parse_json() (*propertyestimator.workflow.schemas.ProtocolReplicator* class method), 88
 parse_json() (*propertyestimator.workflow.schemas.ProtocolSchema* class method), 86
 parse_json() (*propertyestimator.workflow.schemas.WorkflowSchema* class method), 85
 per_thread_memory_limit() (*propertyestimator.backends.QueueWorkerResources* property), 69
 perturbation_scale (*propertyestimator.protocols.gradients.GradientReducedPotentials* attribute), 199
 PhysicalProperty (class in *propertyestimator.properties*), 28
 PhysicalPropertyDataSet (class in *propertyestimator.datasets*), 53
 placeholder_id() (*propertyestimator.workflow.schemas.ProtocolReplicator* property), 88
 PlaceholderInput (class in *propertyestimator.workflow.utils*), 94
 pop_next_in_path() (*propertyestimator.workflow.utils.ProtocolPath* method), 96
 preferred_charge_model() (*propertyestimator.forcefield.LigParGenForceFieldSource* property), 23
 preferred_gpu_toolkit() (*propertyestimator.backends.ComputeResources* property), 67
 preferred_gpu_toolkit() (*propertyestimator.backends.QueueWorkerResources* property), 69
 prepend_protocol_id() (*propertyestimator.workflow.utils.ProtocolPath* method), 96
 pressure (*propertyestimator.thermodynamics.ThermodynamicState* attribute), 51
 pressure() (*propertyestimator.properties.Density* property), 35
 pressure() (*propertyestimator.properties.DielectricConstant* property), 39
 pressure() (*propertyestimator.properties.EnthalpyOfMixing* property), 41
 pressure() (*propertyestimator.properties.EnthalpyOfVaporization* property), 44
 pressure() (*propertyestimator.properties.ExcessMolarVolume* property), 37
 pressure() (*propertyestimator.properties.HostGuestBindingAffinity* property), 46
 pressure() (*propertyestimator*

- tor.properties.PhysicalProperty* (*property*), 29
- production_simulation()* (*propertyestimator.protocols.utils.BaseSimulationProtocols* *property*), 250
- properties* (*propertyestimator.client.PropertyEstimatorSubmission* *attribute*), 15
- properties()* (*propertyestimator.datasets.PhysicalPropertyDataSet* *property*), 53
- properties()* (*propertyestimator.datasets.ThermoMLDataSet* *property*), 61
- property_name()* (*propertyestimator.workflow.utils.ProtocolPath* *property*), 96
- PropertyCalculationLayer* (*class in propertyestimator.layers*), 63
- PropertyEstimatorBackend* (*class in propertyestimator.backends*), 66
- PropertyEstimatorClient* (*class in propertyestimator.client*), 10
- PropertyEstimatorClient.Request* (*class in propertyestimator.client*), 13
- PropertyEstimatorOptions* (*class in propertyestimator.client*), 14
- PropertyEstimatorResult* (*class in propertyestimator.client*), 16
- PropertyEstimatorServer* (*class in propertyestimator.server*), 25
- PropertyEstimatorServer.ServerEstimationRequest* (*class in propertyestimator.server*), 26
- PropertyEstimatorStorage* (*class in propertyestimator.storage*), 73
- PropertyEstimatorSubmission* (*class in propertyestimator.client*), 15
- PropertyPhase* (*class in propertyestimator.properties*), 30
- protocol_input* (*in module propertyestimator.workflow.decorators*), 97
- protocol_output* (*in module propertyestimator.workflow.decorators*), 97
- protocol_path()* (*propertyestimator.workflow.utils.ProtocolPath* *property*), 96
- ProtocolGroup* (*class in propertyestimator.protocols.groups*), 207
- ProtocolGroupSchema* (*class in propertyestimator.workflow.schemas*), 86
- ProtocolPath* (*class in propertyestimator.workflow.utils*), 95
- ProtocolReplicator* (*class in propertyestimator.workflow.schemas*), 87
- protocols()* (*propertyestimator.protocols.groups.ConditionalGroup* *property*), 215
- protocols()* (*propertyestimator.protocols.groups.ProtocolGroup* *property*), 208
- ProtocolSchema* (*class in propertyestimator.workflow.schemas*), 85
- provenance* (*propertyestimator.properties.CalculationSource* *attribute*), 32
- provenance* (*propertyestimator.storage.dataclasses.BaseStoredData* *attribute*), 77
- ## Q
- queued_properties* (*propertyestimator.client.PropertyEstimatorResult* *attribute*), 17
- QueueWorkerResources* (*class in propertyestimator.backends*), 68
- QueueWorkerResources.GPUSoftwareToolkit* (*class in propertyestimator.backends*), 69
- ## R
- receptor_coordinate_file* (*propertyestimator.protocols.coordinates.BuildDockedCoordinates* *attribute*), 109
- receptor_residue_name* (*propertyestimator.protocols.coordinates.BuildDockedCoordinates* *attribute*), 110
- receptor_residue_name* (*propertyestimator.protocols.yank.LigandReceptorYankProtocol* *attribute*), 138
- reduced_reference_potential()* (*propertyestimator.protocols.utils.BaseReweightingProtocols* *property*), 248
- reduced_target_potential()* (*propertyestimator.protocols.utils.BaseReweightingProtocols* *property*), 248
- reference* (*propertyestimator.properties.MeasurementSource* *attribute*), 31
- reference_force_field_paths* (*propertyestimator.protocols.gradients.GradientReducedPotentials* *attribute*), 198
- reference_potential_paths* (*propertyestimator.protocols.gradients.GradientReducedPotentials* *attribute*), 199
- reference_reduced_potentials* (*propertyestimator*

<i>tor.protocols.reweighting.BaseMBARProtocol</i> attribute), 188	<i>replace_protocol()</i> (<i>propertyestimator.protocols.groups.ProtocolGroup</i> method), 208
<i>reference_reduced_potentials</i> (<i>propertyestimator.protocols.reweighting.ReweightStatistics</i> attribute), 195	<i>replace_protocol()</i> (<i>propertyestimator.protocols.miscellaneous.AddValues</i> method), 227
<i>reference_statistics_path</i> (<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> attribute), 198	<i>replace_protocol()</i> (<i>propertyestimator.protocols.miscellaneous.DivideValue</i> method), 238
<i>register_calculation_layer()</i> (in module <i>propertyestimator.layers</i>), 64	<i>replace_protocol()</i> (<i>propertyestimator.protocols.miscellaneous.FilterSubstanceByRole</i> method), 242
<i>register_thermoml_property()</i> (in module <i>propertyestimator.datasets</i>), 62	<i>replace_protocol()</i> (<i>propertyestimator.protocols.miscellaneous.MultiplyValue</i> method), 235
<i>replace_protocol()</i> (<i>propertyestimator.protocols.analysis.AveragePropertyProtocol</i> method), 153	<i>replace_protocol()</i> (<i>propertyestimator.protocols.miscellaneous.SubtractValues</i> method), 231
<i>replace_protocol()</i> (<i>propertyestimator.protocols.analysis.AverageTrajectoryProperty</i> method), 157	<i>replace_protocol()</i> (<i>propertyestimator.protocols.miscellaneous.WeightByMoleFraction</i> method), 246
<i>replace_protocol()</i> (<i>propertyestimator.protocols.analysis.ExtractAverageStatistic</i> method), 161	<i>replace_protocol()</i> (<i>propertyestimator.protocols.reweighting.BaseMBARProtocol</i> method), 191
<i>replace_protocol()</i> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedData</i> method), 165	<i>replace_protocol()</i> (<i>propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM</i> method), 186
<i>replace_protocol()</i> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData</i> method), 173	<i>replace_protocol()</i> (<i>propertyestimator.protocols.reweighting.ConcatenateStatistics</i> method), 181
<i>replace_protocol()</i> (<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData</i> method), 169	<i>replace_protocol()</i> (<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories</i> method), 178
<i>replace_protocol()</i> (<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol</i> method), 103	<i>replace_protocol()</i> (<i>propertyestimator.protocols.reweighting.ReweightStatistics</i> method), 195
<i>replace_protocol()</i> (<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates</i> method), 112	<i>replace_protocol()</i> (<i>propertyestimator.protocols.simulation.RunEnergyMinimisation</i> method), 125
<i>replace_protocol()</i> (<i>propertyestimator.protocols.coordinates.SolvateExistingStructure</i> method), 107	<i>replace_protocol()</i> (<i>propertyestimator.protocols.simulation.RunOpenMMSimulation</i> method), 131
<i>replace_protocol()</i> (<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem</i> method), 116	<i>replace_protocol()</i> (<i>propertyestimator.protocols.storage.UnpackStoredDataCollection</i> method), 219
<i>replace_protocol()</i> (<i>propertyestimator.protocols.forcefield.BuildTLeapSystem</i> method), 121	<i>replace_protocol()</i> (<i>propertyestimator.protocols.storage.UnpackStoredSimulationData</i> method), 223
<i>replace_protocol()</i> (<i>propertyestimator.protocols.gradients.CentralDifferenceGradient</i> method), 206	<i>replace_protocol()</i> (<i>propertyestimator.protocols.yank.BaseYankProtocol</i> method), 136
<i>replace_protocol()</i> (<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> method), 202	<i>replace_protocol()</i> (<i>propertyestimator.protocols.yank.LigandReceptorYankProtocol</i> method), 141
<i>replace_protocol()</i> (<i>propertyestimator.protocols.groups.ConditionalGroup</i> method), 213	

- replace_protocol() (propertyestimator.protocols.yank.SolvationYankProtocol method), 148
- replace_protocol() (propertyestimator.workflow.protocols.BaseProtocol method), 92
- replace_protocol() (propertyestimator.workflow.utils.ProtocolPath method), 96
- replace_protocol() (propertyestimator.workflow.Workflow method), 81
- replace_protocol_types() (propertyestimator.workflow.schemas.WorkflowSchema method), 85
- ReplicatorValue (class in propertyestimator.workflow.utils), 95
- request_estimate() (propertyestimator.client.PropertyEstimatorClient method), 13
- request_url() (propertyestimator.forcefield.LigParGenForceFieldSource property), 24
- required_effective_samples (propertyestimator.protocols.reweighting.BaseMBARProtocol attribute), 188
- required_effective_samples (propertyestimator.protocols.reweighting.ReweightStatistics attribute), 196
- restraint_type (propertyestimator.protocols.yank.LigandReceptorYankProtocol attribute), 139
- result (propertyestimator.protocols.miscellaneous.AddValues attribute), 225
- result (propertyestimator.protocols.miscellaneous.DivideValue attribute), 236
- result (propertyestimator.protocols.miscellaneous.MultiplyValue attribute), 232
- result (propertyestimator.protocols.miscellaneous.SubtractValues attribute), 229
- results() (propertyestimator.client.PropertyEstimatorClient.Request method), 13
- retain_packmol_files (propertyestimator.protocols.coordinates.BuildCoordinatesPackmol attribute), 100
- retain_packmol_files (propertyestimator.protocols.coordinates.SolvateExistingStructure attribute), 107
- retrieve_force_field() (propertyestimator.storage.LocalFileStorage method), 76
- retrieve_force_field() (propertyestimator.storage.PropertyEstimatorStorage method), 73
- retrieve_simulation_data() (propertyestimator.storage.LocalFileStorage method), 76
- retrieve_simulation_data() (propertyestimator.storage.PropertyEstimatorStorage method), 74
- retrieve_simulation_data_by_id() (propertyestimator.storage.LocalFileStorage method), 75
- retrieve_simulation_data_by_id() (propertyestimator.storage.PropertyEstimatorStorage method), 74
- reverse_observable_value (propertyestimator.protocols.gradients.CentralDifferenceGradient attribute), 204
- reverse_parameter_value (propertyestimator.protocols.gradients.CentralDifferenceGradient attribute), 204
- reverse_parameter_value (propertyestimator.protocols.gradients.GradientReducedPotentials attribute), 200
- reverse_potentials_path (propertyestimator.protocols.gradients.GradientReducedPotentials attribute), 199
- ReweightingLayer (class in propertyestimator.layers), 64
- ReweightStatistics (class in propertyestimator.protocols.reweighting), 191
- role() (propertyestimator.substances.Substance.Component property), 48
- root_directory() (propertyestimator.storage.LocalFileStorage property), 75
- root_protocols() (propertyestimator.protocols.groups.ConditionalGroup property), 215
- root_protocols() (propertyestimator.protocols.groups.ProtocolGroup property), 208
- RunEnergyMinimisation (class in propertyestimator.protocols.simulation), 122
- RunOpenMMSimulation (class in propertyestimator.protocols.simulation), 126
- ## S
- schedule_calculation() (propertyestimator.layers.PropertyCalculationLayer static method), 63
- schedule_calculation() (propertyestimator.layers.ReweightingLayer static method), 64

<code>schedule_calculation()</code>	(<i>propertyestimator.layers.SimulationLayer</i> static method), 65	<code>schema()</code>	(<i>propertyestimator.protocols.miscellaneous.FilterSubstanceByRole</i> property), 243
<code>schema()</code>	(<i>propertyestimator.protocols.analysis.AveragePropertyProtocol</i> property), 153	<code>schema()</code>	(<i>propertyestimator.protocols.miscellaneous.MultiplyValue</i> property), 235
<code>schema()</code>	(<i>propertyestimator.protocols.analysis.AverageTrajectoryProperty</i> property), 157	<code>schema()</code>	(<i>propertyestimator.protocols.miscellaneous.SubtractValues</i> property), 231
<code>schema()</code>	(<i>propertyestimator.protocols.analysis.ExtractAverageStatistic</i> property), 162	<code>schema()</code>	(<i>propertyestimator.protocols.miscellaneous.WeightByMoleFraction</i> property), 247
<code>schema()</code>	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedData</i> property), 166	<code>schema()</code>	(<i>propertyestimator.protocols.reweighting.BaseMBARProtocol</i> property), 191
<code>schema()</code>	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData</i> property), 174	<code>schema()</code>	(<i>propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM</i> property), 186
<code>schema()</code>	(<i>propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData</i> property), 170	<code>schema()</code>	(<i>propertyestimator.protocols.reweighting.ConcatenateStatistics</i> property), 182
<code>schema()</code>	(<i>propertyestimator.protocols.coordinates.BuildCoordinatesPackmol</i> property), 103	<code>schema()</code>	(<i>propertyestimator.protocols.reweighting.ConcatenateTrajectories</i> property), 178
<code>schema()</code>	(<i>propertyestimator.protocols.coordinates.BuildDockedCoordinates</i> property), 112	<code>schema()</code>	(<i>propertyestimator.protocols.reweighting.ReweightStatistics</i> property), 196
<code>schema()</code>	(<i>propertyestimator.protocols.coordinates.SolvateExistingStructure</i> property), 107	<code>schema()</code>	(<i>propertyestimator.protocols.simulation.RunEnergyMinimisation</i> property), 126
<code>schema()</code>	(<i>propertyestimator.protocols.forcefield.BuildSmirnoffSystem</i> property), 117	<code>schema()</code>	(<i>propertyestimator.protocols.simulation.RunOpenMMSimulation</i> property), 131
<code>schema()</code>	(<i>propertyestimator.protocols.forcefield.BuildTLeapSystem</i> property), 121	<code>schema()</code>	(<i>propertyestimator.protocols.storage.UnpackStoredDataCollection</i> property), 219
<code>schema()</code>	(<i>propertyestimator.protocols.gradients.CentralDifferenceGradient</i> property), 207	<code>schema()</code>	(<i>propertyestimator.protocols.storage.UnpackStoredSimulationData</i> property), 223
<code>schema()</code>	(<i>propertyestimator.protocols.gradients.GradientReducedPotentials</i> property), 202	<code>schema()</code>	(<i>propertyestimator.protocols.yank.BaseYankProtocol</i> property), 136
<code>schema()</code>	(<i>propertyestimator.protocols.groups.ConditionalGroup</i> property), 215	<code>schema()</code>	(<i>propertyestimator.protocols.yank.LigandReceptorYankProtocol</i> property), 142
<code>schema()</code>	(<i>propertyestimator.protocols.groups.ProtocolGroup</i> property), 211	<code>schema()</code>	(<i>propertyestimator.protocols.yank.SolvationYankProtocol</i> property), 148
<code>schema()</code>	(<i>propertyestimator.protocols.miscellaneous.AddValues</i> property), 227	<code>schema()</code>	(<i>propertyestimator.workflow.protocols.BaseProtocol</i> property), 92
<code>schema()</code>	(<i>propertyestimator.protocols.miscellaneous.DivideValue</i> property), 239	<code>server_address</code>	(<i>propertyestimator.client.ConnectionOptions</i> attribute), 18
		<code>server_address()</code>	(<i>propertyestimator</i>

<i>tor.client.PropertyEstimatorClient.Request</i> <i>property</i>), 13	<i>method</i>), 227
server_port (propertyestimator.client.ConnectionOptions attribute), 18	set_uuid() (propertyestimator.protocols.miscellaneous.DivideValue method), 239
server_port() (propertyestimator.client.PropertyEstimatorClient.Request property), 13	set_uuid() (propertyestimator.protocols.miscellaneous.FilterSubstanceByRole method), 243
set_uuid() (propertyestimator.protocols.analysis.AveragePropertyProtocol method), 153	set_uuid() (propertyestimator.protocols.miscellaneous.MultiplyValue method), 235
set_uuid() (propertyestimator.protocols.analysis.AverageTrajectoryProperty method), 157	set_uuid() (propertyestimator.protocols.miscellaneous.SubtractValues method), 231
set_uuid() (propertyestimator.protocols.analysis.ExtractAverageStatistic method), 162	set_uuid() (propertyestimator.protocols.miscellaneous.WeightByMoleFraction method), 247
set_uuid() (propertyestimator.protocols.analysis.ExtractUncorrelatedData method), 166	set_uuid() (propertyestimator.protocols.reweighting.BaseMBARProtocol method), 191
set_uuid() (propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData method), 174	set_uuid() (propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM method), 186
set_uuid() (propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData method), 170	set_uuid() (propertyestimator.protocols.reweighting.ConcatenateStatistics method), 182
set_uuid() (propertyestimator.protocols.coordinates.BuildCoordinatesPackmol method), 103	set_uuid() (propertyestimator.protocols.reweighting.ConcatenateTrajectories method), 178
set_uuid() (propertyestimator.protocols.coordinates.BuildDockedCoordinates method), 112	set_uuid() (propertyestimator.protocols.reweighting.ReweightStatistics method), 196
set_uuid() (propertyestimator.protocols.coordinates.SolvateExistingStructure method), 108	set_uuid() (propertyestimator.protocols.simulation.RunEnergyMinimisation method), 126
set_uuid() (propertyestimator.protocols.forcefield.BuildSmirnoffSystem method), 117	set_uuid() (propertyestimator.protocols.simulation.RunOpenMMSimulation method), 131
set_uuid() (propertyestimator.protocols.forcefield.BuildTLeapSystem method), 121	set_uuid() (propertyestimator.protocols.storage.UnpackStoredDataCollection method), 219
set_uuid() (propertyestimator.protocols.gradients.CentralDifferenceGradient method), 207	set_uuid() (propertyestimator.protocols.storage.UnpackStoredSimulationData method), 223
set_uuid() (propertyestimator.protocols.gradients.GradientReducedPotentials method), 202	set_uuid() (propertyestimator.protocols.yank.BaseYankProtocol method), 136
set_uuid() (propertyestimator.protocols.groups.ConditionalGroup method), 213	set_uuid() (propertyestimator.protocols.yank.LigandReceptorYankProtocol method), 142
set_uuid() (propertyestimator.protocols.groups.ProtocolGroup method), 208	set_uuid() (propertyestimator.protocols.yank.SolvationYankProtocol method), 148
set_uuid() (propertyestimator.protocols.miscellaneous.AddValues	set_uuid() (propertyestimator.workflow.protocols.BaseProtocol method),

92			
set_value()	(propertyestimator.properties.Density method), 35	set_value()	(propertyestimator.protocols.gradients.CentralDifferenceGradient method), 207
set_value()	(propertyestimator.properties.DielectricConstant method), 39	set_value()	(propertyestimator.protocols.gradients.GradientReducedPotentials method), 202
set_value()	(propertyestimator.properties.EnthalpyOfMixing method), 41	set_value()	(propertyestimator.protocols.groups.ConditionalGroup method), 213
set_value()	(propertyestimator.properties.EnthalpyOfVaporization method), 44	set_value()	(propertyestimator.protocols.groups.ProtocolGroup method), 210
set_value()	(propertyestimator.properties.ExcessMolarVolume method), 37	set_value()	(propertyestimator.protocols.miscellaneous.AddValues method), 227
set_value()	(propertyestimator.properties.HostGuestBindingAffinity method), 46	set_value()	(propertyestimator.protocols.miscellaneous.DivideValue method), 239
set_value()	(propertyestimator.properties.PhysicalProperty method), 29	set_value()	(propertyestimator.protocols.miscellaneous.FilterSubstanceByRole method), 243
set_value()	(propertyestimator.protocols.analysis.AveragePropertyProtocol method), 153	set_value()	(propertyestimator.protocols.miscellaneous.MultiplyValue method), 235
set_value()	(propertyestimator.protocols.analysis.AverageTrajectoryProperty method), 157	set_value()	(propertyestimator.protocols.miscellaneous.SubtractValues method), 231
set_value()	(propertyestimator.protocols.analysis.ExtractAverageStatistic method), 162	set_value()	(propertyestimator.protocols.miscellaneous.WeightByMoleFraction method), 247
set_value()	(propertyestimator.protocols.analysis.ExtractUncorrelatedData method), 166	set_value()	(propertyestimator.protocols.reweighting.BaseMBARProtocol method), 191
set_value()	(propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData method), 174	set_value()	(propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM method), 186
set_value()	(propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData method), 170	set_value()	(propertyestimator.protocols.reweighting.ConcatenateStatistics method), 182
set_value()	(propertyestimator.protocols.coordinates.BuildCoordinatesPackmol method), 103	set_value()	(propertyestimator.protocols.reweighting.ConcatenateTrajectories method), 178
set_value()	(propertyestimator.protocols.coordinates.BuildDockedCoordinates method), 112	set_value()	(propertyestimator.protocols.reweighting.ReweightStatistics method), 196
set_value()	(propertyestimator.protocols.coordinates.SolvateExistingStructure method), 108	set_value()	(propertyestimator.protocols.simulation.RunEnergyMinimisation method), 126
set_value()	(propertyestimator.protocols.forcefield.BuildSmirnoffSystem method), 117	set_value()	(propertyestimator.protocols.simulation.RunOpenMMSimulation method), 131
set_value()	(propertyestimator.protocols.forcefield.BuildTLeapSystem method), 121	set_value()	(propertyestimator.protocols.storage.UnpackStoredDataCollection method), 219

`set_value()` (*propertyestimator.protocols.storage.UnpackStoredSimulationData* attribute), 223
`set_value()` (*propertyestimator.protocols.yank.BaseYankProtocol* method), 136
`set_value()` (*propertyestimator.protocols.yank.LigandReceptorYankProtocol* method), 142
`set_value()` (*propertyestimator.protocols.yank.SolvationYankProtocol* method), 148
`set_value()` (*propertyestimator.workflow.protocols.BaseProtocol* method), 93
`setup_only` (*propertyestimator.protocols.yank.BaseYankProtocol* attribute), 134
`setup_only` (*propertyestimator.protocols.yank.LigandReceptorYankProtocol* attribute), 142
`setup_only` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 148
`simulation_data_path` (*propertyestimator.protocols.storage.UnpackStoredSimulationData* attribute), 220
`SimulationLayer` (class in *propertyestimator.layers*), 65
`smiles()` (*propertyestimator.substances.Substance.Component* property), 48
`SmirnoffForceFieldSource` (class in *propertyestimator.forcefield*), 20
`solute` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 144
`solute_coordinate_file` (*propertyestimator.protocols.coordinates.SolvateExistingStructure* attribute), 104
`solvated_complex_coordinates` (*propertyestimator.protocols.yank.LigandReceptorYankProtocol* attribute), 138
`solvated_complex_system` (*propertyestimator.protocols.yank.LigandReceptorYankProtocol* attribute), 139
`solvated_complex_trajectory_path` (*propertyestimator.protocols.yank.LigandReceptorYankProtocol* attribute), 139
`solvated_ligand_coordinates` (*propertyestimator.protocols.yank.LigandReceptorYankProtocol* attribute), 138
`solvated_ligand_system` (*propertyestimator.protocols.yank.LigandReceptorYankProtocol* attribute), 138
`solvated_ligand_trajectory_path` (*propertyestimator.protocols.yank.LigandReceptorYankProtocol* attribute), 139
`SolvateExistingStructure` (class in *propertyestimator.protocols.coordinates*), 103
`SolvationYankProtocol` (class in *propertyestimator.protocols.yank*), 143
`solvent_1` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 144
`solvent_1_coordinates` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 144
`solvent_1_system` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 145
`solvent_1_trajectory_path` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 145
`solvent_2` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 144
`solvent_2_coordinates` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 145
`solvent_2_system` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 145
`solvent_2_trajectory_path` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 145
`Source` (class in *propertyestimator.properties*), 30
`source_calculation_id` (*propertyestimator.storage.dataclasses.BaseStoredData* attribute), 77
`sources()` (*propertyestimator.datasets.PhysicalPropertyDataSet* property), 54
`sources()` (*propertyestimator.datasets.ThermoMLDataSet* property), 62
`start()` (*propertyestimator.backends.BaseDaskBackend* method), 70
`start()` (*propertyestimator.backends.DaskLocalCluster* method), 70
`start()` (*propertyestimator.backends.DaskLSFBackend* method),

72

`start()` (*propertyestimator.backends.PropertyEstimatorBackend* method), 66

`start()` (*propertyestimator.server.PropertyEstimatorServer* method), 28

`start_listening_loop()` (*propertyestimator.server.PropertyEstimatorServer* method), 27

`start_protocol()` (*propertyestimator.workflow.utils.ProtocolPath* property), 96

`statistical_inefficiency` (*propertyestimator.protocols.analysis.AveragePropertyProtocol* attribute), 151

`statistical_inefficiency` (*propertyestimator.protocols.analysis.AverageTrajectoryProperty* attribute), 158

`statistical_inefficiency` (*propertyestimator.protocols.analysis.ExtractAverageStatistic* attribute), 162

`statistical_inefficiency` (*propertyestimator.protocols.analysis.ExtractUncorrelatedData* attribute), 163

`statistical_inefficiency` (*propertyestimator.protocols.analysis.ExtractUncorrelatedStatisticsData* attribute), 174

`statistical_inefficiency` (*propertyestimator.protocols.analysis.ExtractUncorrelatedTrajectoryData* attribute), 170

`statistical_inefficiency` (*propertyestimator.protocols.storage.UnpackStoredSimulationData* attribute), 221

`statistical_inefficiency` (*propertyestimator.storage.dataclasses.StoredSimulationData* attribute), 78

`statistical_inefficiency` (*propertyestimator.workflow.schemas.WorkflowSimulationDataToStore* attribute), 89

`statistics_file_name` (*propertyestimator.storage.dataclasses.StoredSimulationData* attribute), 78

`statistics_file_path` (*propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM* attribute), 184

`statistics_file_path` (*propertyestimator.protocols.simulation.RunOpenMMSimulation* attribute), 129

`statistics_file_path` (*propertyestimator.protocols.storage.UnpackStoredSimulationData* attribute), 221

`statistics_file_path` (*propertyestimator.workflow.schemas.WorkflowSimulationDataToStore* attribute), 89

`statistics_path` (*propertyestimator.protocols.analysis.ExtractAverageStatistic* attribute), 159

`statistics_paths` (*propertyestimator.protocols.reweighting.ReweightStatistics* attribute), 193

`statistics_type` (*propertyestimator.protocols.analysis.ExtractAverageStatistic* attribute), 159

`statistics_type` (*propertyestimator.protocols.reweighting.ReweightStatistics* attribute), 193

`steps_per_iteration` (*propertyestimator.protocols.simulation.RunOpenMMSimulation* attribute), 127

`steps_per_iteration` (*propertyestimator.protocols.yank.BaseYankProtocol* attribute), 133

`steps_per_iteration` (*propertyestimator.protocols.yank.LigandReceptorYankProtocol* attribute), 142

`steps_per_iteration` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 148

`steric_lambdas_1` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 145

`steric_lambdas_2` (*propertyestimator.protocols.yank.SolvationYankProtocol* attribute), 145

`stop()` (*propertyestimator.backends.BaseDaskBackend* method), 70

`stop()` (*propertyestimator.backends.DaskLocalCluster* method), 71

`stop()` (*propertyestimator.backends.DaskLSFBackend* method), 72

`stop()` (*propertyestimator.backends.PropertyEstimatorBackend* method), 66

`stop()` (*propertyestimator.server.PropertyEstimatorServer* method), 27

`store_force_field()` (*propertyestimator.storage.LocalFileStorage* method), 76

`store_force_field()` (*propertyestimator.storage.PropertyEstimatorStorage* method), 73

`store_simulation_data()` (*propertyestimator.storage.LocalFileStorage* method), 75

`store_simulation_data()` (*propertyestimator.storage.PropertyEstimatorStorage* method), 74

`StoredDataCollection` (class in *propertyestimator*), 74

- tor.storage.dataclasses*), 79
- StoredSimulationData (class in *propertyestimator.storage.dataclasses*), 78
- submit () (*propertyestimator.workflow.WorkflowGraph* method), 83
- submit_task () (*propertyestimator.backends.BaseDaskBackend* method), 70
- submit_task () (*propertyestimator.backends.DaskLocalCluster* method), 70
- submit_task () (*propertyestimator.backends.DaskLSFBackend* method), 72
- submit_task () (*propertyestimator.backends.PropertyEstimatorBackend* method), 66
- Substance (class in *propertyestimator.substances*), 46
- substance (*propertyestimator.protocols.coordinates.BuildCoordinatesPackmol* attribute), 100
- substance (*propertyestimator.protocols.coordinates.SolvateExistingStructure* attribute), 108
- substance (*propertyestimator.protocols.forcefield.BuildSmirnoffSystem* attribute), 117
- substance (*propertyestimator.protocols.forcefield.BuildTLeapSystem* attribute), 121
- substance (*propertyestimator.protocols.gradients.GradientReducedPotential* attribute), 199
- substance (*propertyestimator.protocols.storage.UnpackStoredSimulationData* attribute), 220
- substance (*propertyestimator.storage.dataclasses.BaseStoredData* attribute), 77
- substance (*propertyestimator.workflow.schemas.WorkflowOutputToStore* attribute), 89
- Substance.Amount (class in *propertyestimator.substances*), 48
- Substance.Component (class in *propertyestimator.substances*), 48
- Substance.ComponentRole (class in *propertyestimator.substances*), 48
- Substance.ExactAmount (class in *propertyestimator.substances*), 49
- Substance.MoleFraction (class in *propertyestimator.substances*), 49
- SubtractValues (class in *propertyestimator.protocols.miscellaneous*), 228
- system_path (*propertyestimator.protocols.forcefield.BuildSmirnoffSystem* attribute), 117
- system_path (*propertyestimator.protocols.forcefield.BuildTLeapSystem* attribute), 121
- system_path (*propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM* attribute), 183
- system_path (*propertyestimator.protocols.simulation.RunEnergyMinimisation* attribute), 123
- system_path (*propertyestimator.protocols.simulation.RunOpenMMSimulation* attribute), 128
- ## T
- target_reduced_potentials (*propertyestimator.protocols.reweighting.BaseMBARProtocol* attribute), 188
- target_reduced_potentials (*propertyestimator.protocols.reweighting.ReweightStatistics* attribute), 196
- temperature (*propertyestimator.thermodynamics.ThermodynamicState* attribute), 51
- temperature () (*propertyestimator.properties.Density* property), 35
- temperature () (*propertyestimator.properties.DielectricConstant* property), 39
- temperature () (*propertyestimator.properties.EnthalpyOfMixing* property), 42
- temperature () (*propertyestimator.properties.EnthalpyOfVaporization* property), 44
- temperature () (*propertyestimator.properties.ExcessMolarVolume* property), 37
- temperature () (*propertyestimator.properties.HostGuestBindingAffinity* property), 46
- temperature () (*propertyestimator.properties.PhysicalProperty* property), 29
- thermodynamic_state (*propertyestimator.protocols.gradients.GradientReducedPotentials* attribute), 199
- thermodynamic_state (*propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM* attribute), 183
- thermodynamic_state (*propertyestimator.protocols.simulation.RunOpenMMSimulation* attribute), 128

- attribute*), 128
- thermodynamic_state (*propertyestimator.protocols.storage.UnpackStoredSimulationData attribute*), 220
- thermodynamic_state (*propertyestimator.protocols.yank.BaseYankProtocol attribute*), 133
- thermodynamic_state (*propertyestimator.protocols.yank.LigandReceptorYankProtocol attribute*), 142
- thermodynamic_state (*propertyestimator.protocols.yank.SolvationYankProtocol attribute*), 149
- thermodynamic_state (*propertyestimator.storage.dataclasses.BaseStoredData attribute*), 77
- ThermodynamicState (*class in propertyestimator.thermodynamics*), 51
- ThermoMLDataSet (*class in propertyestimator.datasets*), 57
- thermostat_friction (*propertyestimator.protocols.simulation.RunOpenMMSimulation attribute*), 128
- timestep (*propertyestimator.protocols.simulation.RunOpenMMSimulation attribute*), 128
- timestep (*propertyestimator.protocols.yank.BaseYankProtocol attribute*), 133
- timestep (*propertyestimator.protocols.yank.LigandReceptorYankProtocol attribute*), 142
- timestep (*propertyestimator.protocols.yank.SolvationYankProtocol attribute*), 149
- TLeapForceFieldSource (*class in propertyestimator.forcefield*), 21
- to_components () (*propertyestimator.workflow.utils.ProtocolPath static method*), 96
- to_force_field () (*propertyestimator.forcefield.SmirnoffForceFieldSource method*), 20
- to_number_of_molecules () (*propertyestimator.substances.Substance.Amount method*), 48
- to_number_of_molecules () (*propertyestimator.substances.Substance.ExactAmount method*), 49
- to_number_of_molecules () (*propertyestimator.substances.Substance.MoleFraction method*), 49
- to_pandas () (*propertyestimator.datasets.PhysicalPropertyDataSet method*), 56
- to_pandas () (*propertyestimator.datasets.ThermoMLDataSet method*), 62
- tolerance (*propertyestimator.protocols.simulation.RunEnergyMinimisation attribute*), 123
- total_number_of_iterations (*propertyestimator.protocols.simulation.RunOpenMMSimulation attribute*), 127
- total_number_of_molecules (*propertyestimator.protocols.storage.UnpackStoredSimulationData attribute*), 220
- total_number_of_molecules (*propertyestimator.storage.dataclasses.StoredSimulationData attribute*), 78
- total_number_of_molecules (*propertyestimator.workflow.schemas.WorkflowSimulationDataToStore attribute*), 89
- trajectory_file_name (*propertyestimator.storage.dataclasses.StoredSimulationData attribute*), 78
- trajectory_file_path (*propertyestimator.protocols.gradients.GradientReducedPotentials attribute*), 199
- trajectory_file_path (*propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM attribute*), 183
- trajectory_file_path (*propertyestimator.protocols.simulation.RunOpenMMSimulation attribute*), 129
- trajectory_file_path (*propertyestimator.protocols.storage.UnpackStoredSimulationData attribute*), 221
- trajectory_file_path (*propertyestimator.workflow.schemas.WorkflowSimulationDataToStore attribute*), 89
- trajectory_path (*propertyestimator.protocols.analysis.AverageTrajectoryProperty attribute*), 155
- ## U
- uncorrelated_values (*propertyestimator.protocols.analysis.AveragePropertyProtocol attribute*), 151
- uncorrelated_values (*propertyestimator.protocols.analysis.AverageTrajectoryProperty attribute*), 158
- uncorrelated_values (*propertyestimator.protocols.analysis.ExtractAverageStatistic attribute*), 162
- unpack_stored_data () (*propertyestimator.protocols.utils.BaseReweightingProtocols property*), 249

- UnpackStoredDataCollection (class in *propertyestimator.protocols.storage*), 215
- UnpackStoredSimulationData (class in *propertyestimator.protocols.storage*), 219
- unsuccessful_properties (propertyestimator.client.PropertyEstimatorResult attribute), 17
- update_references() (propertyestimator.workflow.schemas.ProtocolReplicator method), 88
- use_internal_energy (propertyestimator.protocols.reweighting.CalculateReducedPotentialOpenMM3 attribute), 184
- use_subset_of_force_field (propertyestimator.protocols.gradients.GradientReducedPotentials attribute), 199
- ## V
- validate_interfaces() (propertyestimator.workflow.schemas.WorkflowSchema method), 85
- value (propertyestimator.protocols.analysis.AveragePropertyProtocol attribute), 151
- value (propertyestimator.protocols.analysis.AverageTrajectoryProperty attribute), 158
- value (propertyestimator.protocols.analysis.ExtractAverageStatistic attribute), 162
- value (propertyestimator.protocols.miscellaneous.DivideValue attribute), 236
- value (propertyestimator.protocols.miscellaneous.MultiplyValue attribute), 232
- value (propertyestimator.protocols.miscellaneous.WeightByMoleFraction attribute), 244
- value (propertyestimator.protocols.reweighting.BaseMBARProtocol attribute), 188
- value (propertyestimator.protocols.reweighting.ReweightStatistics attribute), 196
- value() (propertyestimator.substances.Substance.Amount property), 48
- value() (propertyestimator.substances.Substance.ExactAmount property), 49
- value() (propertyestimator.substances.Substance.MoleFraction property), 49
- value_a (propertyestimator.protocols.miscellaneous.SubtractValues attribute), 228
- value_b (propertyestimator.protocols.miscellaneous.SubtractValues attribute), 229
- values (propertyestimator.protocols.miscellaneous.AddValues attribute), 225
- verbose (propertyestimator.protocols.yank.BaseYankProtocol attribute), 143
- verbose (propertyestimator.protocols.yank.LigandReceptorYankProtocol attribute), 142
- verbose (propertyestimator.protocols.yank.SolvationYankProtocol attribute), 149
- verbose_packmol (propertyestimator.protocols.coordinates.BuildCoordinatesPackmol attribute), 100
- verbose_packmol (propertyestimator.protocols.coordinates.SolvateExistingStructure attribute), 108
- ## W
- wallclock_time_limit() (propertyestimator.backends.QueueWorkerResources property), 69
- water_model (propertyestimator.protocols.forcefield.BuildSmirnoffSystem attribute), 117
- water_model (propertyestimator.protocols.forcefield.BuildTLeapSystem attribute), 121
- WeightByMoleFraction (class in *propertyestimator.protocols.miscellaneous*), 243
- weighted_value (propertyestimator.protocols.miscellaneous.WeightByMoleFraction attribute), 244
- Workflow (class in *propertyestimator.workflow*), 81
- workflow_options (propertyestimator.client.PropertyEstimatorOptions attribute), 14
- workflow_schemas (propertyestimator.client.PropertyEstimatorOptions attribute), 14
- WorkflowDataCollectionToStore (class in *propertyestimator.workflow.schemas*), 90
- WorkflowGraph (class in *propertyestimator.workflow*), 82
- WorkflowOptions (class in *propertyestimator.workflow*), 83

WorkflowOptions.ConvergenceMode (*class in propertyestimator.workflow*), 84
WorkflowOutputToStore (*class in propertyestimator.workflow.schemas*), 89
WorkflowSchema (*class in propertyestimator.workflow.schemas*), 84
WorkflowSimulationDataToStore (*class in propertyestimator.workflow.schemas*), 89