OpenFF Evaluator Documentation

openff-evaluator

Aug 26, 2021

GETTING STARTED

1	Calcu	ulation Approaches	3			
2 Supported Physical Properties						
	2.1	Installation	6			
	2.2	Architecture	7			
	2.3	Evaluator Client	7			
	2.4	Evaluator Server	9			
	2.5	Tutorial 01 - Loading Data Sets	11			
	2.6	Tutorial 02 - Estimating Data Sets	16			
	2.7	Tutorial 03 - Analysing Data Sets	20			
	2.8	Tutorial 04 - Optimizing Force Fields	23			
	2.9	Property Data Sets	30			
	2.10	ThermoML Archive	33			
	2.11	Таргоот	35			
	2.12	Data Set Curation	36			
	2.13	Physical Properties	42			
	2.14	Common Workflows	46			
	2.15	Gradients	48			
	2.16	Calculation Layers	49			
	2.17	Workflow Layers	52			
	2.18	The Direct Simulation Layer	54			
	2.19	The MBAR Reweighting Layer	54			
	2.20	Workflows	55			
	2.21	Replicators	57			
	2.22	Workflow Graphs	61			
	2.23	Protocols	62			
	2.24	Protocol Groups	65			
	2.25	Observables	66			
	2.26	Calculation Backends	68			
	2.27	Dask Backends	69			
	2.28	Storage Backends	71			
	2.29	Data Classes and Queries	72			
	2.30	Local File Storage	74			
	2.31	Building the Docs	75			
	2.32	API	75			
	2.33	Release History	572			
	2.34	Release Process	583			

Bibliography

587

Index

An automated and scalable framework for curating, manipulating, and computing data sets of physical properties from molecular simulation and simulation data.

The framework is built around four central ideas:

- **Flexibility:** New physical properties, data sources and calculation approaches are easily added via an extensible plug-in system and a flexible workflow engine.
- Automation: *Physical property measurements* are readily importable from open data sources (such as the NIST ThermoML Archive) through the data set APIs, and automatically calculated using either the built-in or user specified calculation schemas.
- **Scalability:** Calculations are readily scalable from single machines and laptops up to large HPC clusters and supercomputers through seamless integration with libraries such as dask.
- Efficiency: Properties are estimated using the fastest approach available to the framework, whether that be through evaluating a trained surrogate model, re-evaluating cached simulation data, or by running simulations directly.

CHAPTER

CALCULATION APPROACHES

The framework is designed around the idea of allowing multiple calculation approaches for estimating the same set of properties, in addition to estimation directly from molecular simulation, all using a uniform API.

The primary purpose of this is to take advantage of the many techniques exist which are able to leverage data from previous simulations to rapidly estimate sets of properties, such as reweighting cached simulation data, or evaluating surrogate models trained upon cached data. The most rapid approach which may accurately estimate a set of properties is automatically determined by the framework on the fly.

Each approach supported by the framework is implemented as a *calculation layer*. Two such layers are currently supported (although new calculation layers can be readily added via the plug-in system):

- evaluating physical properties directly from molecular simulation using the *SimulationLayer*.
- reprocessing cached simulation data with MBAR reweighting using the ReweightingLayer.

SUPPORTED PHYSICAL PROPERTIES

The framework has built-in support for evaluating a number of *physical properties*, ranging from relatively 'cheap' to compute properties such as liquid densities, up to more computationally demanding properties such as solvation free energies and host-guest binding affinities.

Included for most of these properties is the ability to calculate their derivatives with respect to force field parameters, making the framework ideal for evaluating an objective function and it's gradient as part of a force field optimisation.

framework.							
	Direct Simulation		MBAR Reweighting				
	Supported	Gradients	Supported	Gradients			
Density	\checkmark	\checkmark	\checkmark	\checkmark			
Dielectric Constant	\checkmark	√*	\checkmark	√*			
H _{vaporization}	\checkmark	\checkmark	\checkmark	\checkmark			
H _{mixing}	\checkmark	\checkmark	√*	\checkmark			
V _{excess}	\checkmark	\checkmark	\checkmark	\checkmark			
G _{solvation}	\checkmark	√*	×	×			
G _{host-guest} (beta)	√*	×	×	×			

Table 1: The physical properties which are natively supported by the framework.

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

See the *physical properties overview page* for more details.

2.1 Installation

The OpenFF Evaluator is currently installable either through conda or directly from the source code. Whichever route is chosen, it is recommended to install the framework within a conda environment and allow the conda package manager to install the required and optional 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.

2.1.1 Installation from Conda

To install the openff-evaluator from the conda-forge channel simply run:

```
conda install -c conda-forge openff-evaluator
```

2.1.2 Recommended Dependencies

If you have access to the fantastic OpenEye toolkit we recommend installing this to enable (among many other things) the use of the BuildDockedCoordinates protocol and faster conformer generation / AM1BCC partial charge calculations:

```
conda install -c openeye openeye-toolkits
```

To parameterize systems with the Amber tleap tool using a TLeapForceFieldSource the ambertools package must be installed:

conda install -c conda-forge 'ambertools >=19.0'

2.1.3 Installation from Source

To install the OpenFF Evaluator from source begin by cloning the repository from github:

```
git clone https://github.com/openforcefield/openff-evaluator.git
cd openff-evaluator
```

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

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

Finally, install the estimator itself:

python setup.py develop

2.2 Architecture

The openff-evaluator framework is constructed as a collection of modular components, each performing a specific role within the estimation of physical property data sets. These components are designed to be as extensible as possible, with support for user created plug-ins built into their core.

Fig. 1: An overview of the openff-evaluators modular design. The framework is split into a 'client-side' which handles the curation and preparation of data sets, and a 'server-side' which performs the estimation of the data sets.

The framework is implemented as a *client-server* architecture. This design allows users to spin up *Evaluator Server* instances on whichever compute resources they may have available (from a single machine up to a large HPC cluster), and to which *Evaluator Client* objects may connect to both request that data sets be estimated, and to query and retrieve the results of those requests.

The *client-side* of the framework is predominantly responsible for providing APIs and objects for:

- curating *data sets* of physical properties from open data sources.
- specifing custom *calculation schemas* which describe how individual properties should be computed.
- requesting that data sets be estimated by a running Evaluator Server instance.
- retrieving the results of estimation requests from a running Evaluator Server instance.

while the *server-side* is responsible for:

- receiving estimation requests from an Evaluator Client object.
- automatically determining which *calculation approach* to use for each property in the request.
- executing those requests across the available *compute resources* following the calculation schemas provided by the client
- caching data from any calculations which may be useful for future calculations.

All communication between servers and clients is handled through the TCP protocol.

2.3 Evaluator Client

The *EvaluatorClient* object is responsible for both submitting requests to estimate a data set of properties to a running *Evaluator Server* instance, and for pulling back the results of those requests when complete.

An *EvaluatorClient* object may optionally be created using a set of *ConnectionOptions* which specifies the network address of the running *Evaluator Server* instance to connect to:

```
# Specify the address of a server running on the local machine.
connection_options = ConnectionOptions(server_address="localhost", server_port=8000)
# Create the client object
evaluator_client = EvaluatorClient(connection_options)
```

2.3.1 Requesting Estimates

The client can request the estimation of a data set of properties using the *request_estimate()* function:

```
# Specify the data set.
data_set = PhysicalPropertyDataSet()
data_set.add_properties(...)
# Specify the force field source.
force_field = SmirnoffForceFieldSource.from_path("openff-1.0.0.offxml")
# Specify some estimation options (optional).
options = client.default_request_options(data_set, force_field)
# Specify the parameters to differentiate with respect to (optional).
gradient_keys = [
   ParameterGradientKey(tag="vdW", smirks="[#6X4:1]", attribute="epsilon")
1
# Request the estimation of the data set.
request, errors = evaluator_client.request_estimate(
   data_set.
    force_field,
   options,
    gradient_keys
)
```

A request must at minimum specify:

- the *data set* of physical properties to estimate.
- the *force field parameters* to estimate the data set using.

and may also optionally specify:

- the options to use when estimating the property set.
- the parameters to differentiate each physical property estimate with respect to.

Note: Gradients can currently only be computed for requests using a SMIRNOFF based force field.

The *request_estimate()* function returns back two objects:

- a Request object which can be used to retrieve the results of the request and,
- an EvaluatorException object which will be populated if any errors occured while submitting the request.

The *Request* object is similar to a Future object, in that it is an object which can be used to query the current status of a request either asynchronously:

results = request.results(synchronous=False)

or synchronously:

results = request.results(synchronous=True)

The results (which may currently be incomplete) are returned back as a *RequestResult* object.

The *Request* object is fully JSON serializable:

```
# Save the request to JSON
request.json(file_path="request.json", format=True)
# Load the request from JSON
request = Request.from_json(file_path="request.json")
```

making it easy to keep track of any open requests.

2.3.2 Request Options

The *RequestOptions* object allows greater control over how properties are estimated by the server. It currently allows control over:

- *calculation_layers*: The *calculation layers* which the server should attempt to use when estimating the data set. The order which the layers are specified in this list is the order which the server will attempt to use each layer.
- *calculation_schemas*: The *calculation schemas* to use for each allowed calculation layer per class of property. These will be automatically populated in the cases where no user specified schema is provided, and where a default schema has been registered with the plugin system for the particular layer and property type.

If no options are passed to *request_estimate()* a default set will be generated through a call to *default_request_options()*. For more information about how default calculation schemas are registered, see the *Default Schemas* section.

2.3.3 Force Field Sources

Different force field representations (e.g. SMIRNOFF, TLeap, LigParGen) are defined within the framework as *ForceFieldSource* objects. A force field source should specify *all* of the options which would be required by a particular force field, such as the non-bonded cutoff or the charge scheme if not specified directly in the force field itself.

Currently the framework has built in support for force fields applied via:

- the OpenFF toolkit (SmirnoffForceFieldSource).
- the tleap program from the AmberTools suite (LigParGenForceFieldSource).
- an instance of the LigParGen server (LigParGenForceFieldSource).

The client will automatically adapt any of the built-in calculation schemas which are based off of the *WorkflowCalculationSchema* to use the correct workflow protocol (*BuildSmirnoffSystem*, *BuildTLeapSystem*) or *BuildLigParGenSystem*) for the requested force field.

2.4 Evaluator Server

The *EvaluatorServer* object is responsible for coordinating the estimation of physical property data sets as requested by *evaluator clients*. Its primary responsibilities are to:

- recieve incoming requests from an *evaluator clients* to either estimate a dataset of properties, or to query the status of a previous request.
- request that each specified *calculation layers* attempt to estimate the data set of properties, cascading unestimated properties through the different layers.

An *EvaluatorServer* must be created with an accompanying *calculation backend* which will be responsible for distributing any calculations launched by the different calculation layers:

```
with DaskLocalCluster() as calculation_backend:
```

```
evaluator_server = EvaluatorServer(calculation_backend)
evaluator_server.start()
```

It may also be optionally created using a specific *storage backend* if the default *LocalFileStorage* is not sufficient:

```
with DaskLocalCluster() as calculation_backend:
    storage_backend = LocalFileStorage()
    evaluator_server = EvaluatorServer(calculation_backend, storage_backend)
    evaluator_server.start()
```

By default the server will run synchronously until it is killed, however it may also be run asynchronously such that it can be interacted with directly by a client in the same script:

```
with DaskLocalCluster() as calculation_backend:
    with EvaluatorServer(calculation_backend) as evaluator_server:
        # Specify the data set.
        data_set = PhysicalPropertyDataSet()
        data_set.add_properties(...)
        # Specify the force field source.
        force_field = SmirnoffForceFieldSource.from_path("openff-1.0.0.offxml")
        # Request the estimation of the data set.
        request, errors = evaluator_client.request_estimate(data_set,force_field)
        # Wait for the results.
        results = request.results(synchronous=True)
```

2.4.1 Estimation Batches

When a server recieves a request from a client, it will attempt to split the requested set of properties into smaller batches, represented by the *Batch* object. The server is currently only able to mark entire batches of estimated properties as being completed, as opposed to individual properties.

Currently the server supports two ways of batching properties:

- SameComponents: All properties measured for the substance containing the *same* components will be batched together. As an example, the density of a 80:20 and a 20:80 mix of ethanol and water would be batched together, but the density of pure ethanol and the density of pure water would be placed into separate batches.
- SharedComponents: All properties measured for substances containing at least one common component will be batched together. As an example, the densities of 80:20 and 20:80 mixtures of ethanol and water, and the pure densities of ethanol and water would be batched together.

The mode of batching is set by the client using the *batch_mode* attribute of the request options.

2.5 Tutorial 01 - Loading Data Sets

In this tutorial we will be exploring the frameworks utilities for loading and manipulating data sets of physical property measurements. The tutorial will cover

- · Loading a data set of density measurements from NISTs ThermoML Archive
- Filtering the data set down using a range of criteria, including temperature pressure, and composition.
- Supplementing the data set with enthalpy of vaporization (ΔH_v) data sourced directly from the literature

If you haven't yet installed the OpenFF Evaluator framework on your machine, check out the *installation instructions here*!

Note: If you are running this tutorial in google colab you will need to run a setup script instead of following the installation instructions:

For the sake of clarity all warnings will be disabled in this tutorial:

```
[2]: import warnings
warnings.filterwarnings('ignore')
import logging
logging.getLogger("openff.toolkit").setLevel(logging.ERROR)
```

2.5.1 Extracting Data from ThermoML

For anyone who is not familiar with the ThermoML archive - it is a fantastic database of physical property measurements which have been extracted from data published in the

- Journal of Chemical and Engineering Data
- · Journal of Chemical Thermodynamics
- Fluid Phase Equilibria
- Thermochimica Acta
- International Journal of Thermophysics

journals. It includes data for a wealth of different physical properties, from simple densities and melting points, to activity coefficients and osmotic coefficients, all of which is freely available. As such, it serves as a fantastic resource for benchmarking and optimising molecular force fields against.

The Evaluator framework has built-in support for extracting this wealth of data, storing the data in easy to manipulate python objects, and for automatically re-computing those properties using an array of calculation techniques, such as molecular simulations and, in future, from trained surrogate models.

This support is provided by the ThermoMLDataSet object:

[3]: from openff.evaluator.datasets.thermoml import ThermoMLDataSet

The ThermoMLDataSet object offers two main routes for extracting data the the archive:

• extracting data directly from the NIST ThermoML web server

· extracting data from a local ThermoML XML archive file

Here we will be extracting data directly from the web server. To pull data from the web server we need to specify the digital object identifiers (DOIs) of the data we wish to extract - these correspond to the DOI of the publication that the data was initially sourced from.

For this tutorial we will be extracting data using the following DOIs:

```
[4]: data_set = ThermoMLDataSet.from_doi(
    "10.1016/j.fluid.2013.10.034",
    "10.1021/je1013476",
)
```

We can inspect the data set to see how many properties were loaded:

```
[5]: len(data_set)
```

[5]: 275

and for how many different substances those properties were measured for:

```
[6]: len(data_set.substances)
```

[6]: 254

We can also easily check which types of properties were loaded in:

```
[7]: print(data_set.property_types)
```

```
{'EnthalpyOfMixing', 'Density'}
```

2.5.2 Filtering the Data Set

The data set object we just created contains many different functions which will allow us to filter the data down, retaining only those measurements which are of interest to us.

The first thing we will do is filter out all of the measurements which aren't density measurements:

```
[8]: from openff.evaluator.datasets.curation.components.filtering import (
    FilterByPropertyTypes,
    FilterByPropertyTypesSchema
)
data_set = FilterByPropertyTypes.apply(
    data_set, FilterByPropertyTypesSchema(property_types=["Density"])
)
print(data_set.property_types)
{'Density'}
```

Next we will filter out all measurements which were made away from atmospheric conditions:

```
[9]: from openff.evaluator.datasets.curation.components.filtering import (
    FilterByPressure,
    FilterByPressureSchema,
```

(continues on next page)

(continued from previous page)

```
FilterByTemperature,
   FilterByTemperatureSchema,
)
print(f"There were {len(data_set)} properties before filtering")
# First filter by temperature.
data_set = FilterByTemperature.apply(
   data_set.
   FilterByTemperatureSchema(minimum_temperature=298.0, maximum_temperature=298.2)
)
# and then by pressure
data_set = FilterByPressure.apply(
    data_set,
   FilterByPressureSchema(minimum_pressure=101.224, maximum_pressure=101.426)
)
print(f"There are now {len(data_set)} properties after filtering")
There were 213 properties before filtering
There are now 9 properties after filtering
```

Finally, we will filter out all measurements which were not measured for either ethanol (CCO) or isopropanol (CC(C)O):

```
[10]: from openff.evaluator.datasets.curation.components.filtering import (
    FilterBySmiles,
    FilterBySmilesSchema,
)
data_set = FilterBySmiles.apply(
    data_set,
    FilterBySmilesSchema(smiles_to_include=["CCO", "CC(C)0"])
)
print(f"There are now {len(data_set)} properties after filtering")
There are now 2 properties after filtering
```

We will convert the filtered data to a pandas DataFrame to more easily visualize the final data set:

```
[11]: pandas_data_set = data_set.to_pandas()
      pandas_data_set[
          ["Temperature (K)", "Pressure (kPa)", "Component 1", "Density Value (g / ml)",
      \rightarrow "Source"]
      ].head()
[11]:
         Temperature (K) Pressure (kPa) Component 1 Density Value (g / ml)
                                                                                298.15
                                  101.325
                                                                       0.78270
      0
                                               CC(C)0
      1
                  298.15
                                  101.325
                                                  CC0
                                                                        0.78507
                               Source
        10.1016/j.fluid.2013.10.034
      0
      1
                   10.1021/je1013476
```

Through filtering, we have now cut down from over 250 property measurements down to just 2. There are many more

possible filters which can be applied. All of these and more information about the data set object can be found in the PhysicalPropertyDataSet (from which the ThermoMLDataSet class inherits) API documentation.

2.5.3 Adding Extra Data

For the final part of this tutorial, we will be supplementing our newly filtered data set with some enthalpy of vaporization (ΔH_v) measurements sourced directly from the literature (as opposed to from the ThermoML archive).

We will be sourcing values of the ΔH_v of ethanol and isopropanol, summarised in the table below, from the Enthalpies of vaporization of some aliphatic alcohols publication:

Compound	Temperature / K	ΔH_v / $kJmol^{-1}$	$\delta\Delta H_v$ / $kJmol^{-1}$
Ethanol	298.15	42.26	0.02
Isopropanol	298.15	45.34	0.02

In order to create a new ΔH_v measurements, we will first define the state (namely temperature and pressure) that the measurements were recorded at:

```
[12]: from openff.evaluator import unit
from openff.evaluator.thermodynamics import ThermodynamicState
thermodynamic_state = ThermodynamicState(
    temperature=298.15 * unit.kelvin, pressure=1.0 * unit.atmosphere
)
```

Note: Here we have made use of the ``openff.evaluator.unit`` module to attach units to the temperatures and pressures we are filtering by. This module simply exposes a ``UnitRegistry`` from the fantasticpintlibrary. Pint provides full support for attaching to units to values and is used extensively throughout this framework.

the substances that the measurements were recorded for:

```
[13]: from openff.evaluator.substances import Substance
```

ethanol = Substance.from_components("CCO")
isopropanol = Substance.from_components("CC(C)O")

and the source of this measurement (defined as the DOI of the publication):

[14]: from openff.evaluator.datasets import MeasurementSource

source = MeasurementSource(doi="10.1016/S0021-9614(71)80108-8")

We will combine this information with the values of the measurements to create an object which encodes each of the ΔH_v measurements

```
[15]: from openff.evaluator.datasets import PropertyPhase
from openff.evaluator.properties import EnthalpyOfVaporization
```

```
ethanol_hvap = EnthalpyOfVaporization(
   thermodynamic_state=thermodynamic_state,
   phase=PropertyPhase.Liquid | PropertyPhase.Gas,
   substance=ethanol,
   value=42.26*unit.kilojoule / unit.mole,
```

(continues on next page)

(continued from previous page)

```
uncertainty=0.02*unit.kilojoule / unit.mole,
source=source
)
isopropanol_hvap = EnthalpyOfVaporization(
    thermodynamic_state=thermodynamic_state,
    phase=PropertyPhase.Liquid | PropertyPhase.Gas,
    substance=isopropanol,
    value=45.34*unit.kilojoule / unit.mole,
    uncertainty=0.02*unit.kilojoule / unit.mole,
    source=source
)
```

These properties can then be added to our data set:

```
[16]: data_set.add_properties(ethanol_hvap, isopropanol_hvap)
```

If we print the data set again using pandas we should see that our new measurements have been added:

```
[17]: pandas_data_set = data_set.to_pandas()
      pandas_data_set[
          ["Temperature (K)",
           "Pressure (kPa)",
           "Component 1",
           "Density Value (g / ml)",
           "EnthalpyOfVaporization Value (kJ / mol)",
           "Source"
           1
      ].head()
         Temperature (K) Pressure (kPa) Component 1 Density Value (g / ml)
[17]:
                                                                                0
                  298.15
                                 101.325
                                               CC(C)0
                                                                       0.78270
                  298.15
                                 101.325
                                                  CC0
                                                                       0.78507
      1
      2
                  298.15
                                 101.325
                                                  CC0
                                                                           NaN
      3
                  298.15
                                 101.325
                                               CC(C)0
                                                                           NaN
         EnthalpyOfVaporization Value (kJ / mol)
                                                                           Source
      0
                                              NaN
                                                     10.1016/j.fluid.2013.10.034
                                                                10.1021/je1013476
      1
                                              NaN
      2
                                            42.26
                                                   10.1016/S0021-9614(71)80108-8
      3
                                            45.34
                                                   10.1016/S0021-9614(71)80108-8
```

2.5.4 Conclusion

We will finish off this tutorial by saving the data set we have created as a JSON file for future use:

```
[18]: data_set.json("filtered_data_set.json", format=True);
```

And that concludes the first tutorial. For more information about data sets in the Evaluator framework check out the *data set* and *ThermoML* documentation.

In the next tutorial we will be estimating the data set we have created here using molecular simulation.

If you have any questions and / or feedback, please open an issue on the GitHub issue tracker.

2.6 Tutorial 02 - Estimating Data Sets

In this tutorial we will be estimating the data set we created in the *first tutorial* using molecular simulation. The tutorial will cover:

- loading in the data set to estimate, and the force field parameters to use in the calculations.
- defining custom calculation schemas for the properties in our data set.
- estimating the data set of properties using an *Evaluator server* instance.
- retrieving the results from the server and storing them on disk.

Note: If you are running this tutorial in google colab you will need to run a setup script instead of following the installation instructions:

For this tutorial make sure that you are using a GPU accelerated runtime.

For the sake of clarity all warnings will be disabled in this tutorial:

```
[2]: import warnings
warnings.filterwarnings('ignore')
import logging
logging.getLogger("openforcefield").setLevel(logging.ERROR)
```

We will also enable time-stamped logging to help track the progress of our calculations:

```
[3]: from openff.evaluator.utils import setup_timestamp_logging
    setup_timestamp_logging()
```

2.6.1 Loading the Data Set and Force Field Parameters

We will begin by loading in the data set which we created in the previous tutorial:

```
[4]: from openff.evaluator.datasets import PhysicalPropertyDataSet
data_set_path = "filtered_data_set.json"
# If you have not yet completed that tutorial or do not have the data set file
# available, a copy is provided by the framework:
# from openff.evaluator.utils import get_data_filename
# data_set_path = get_data_filename("tutorials/tutorial01/filtered_data_set.json")
data_set = PhysicalPropertyDataSet.from_json(data_set_path)
```

As a reminder, this data contains the experimentally measured density and H_{vap} measurements for ethanol and isopropanol at ambient conditions:

[5]: data_set.to_pandas().head()

[5]:		Temperature (K)		Source
	0	298.15		10.1016/j.fluid.2013.10.034
	1	298.15		10.1021/je1013476
	2	298.15		10.1016/S0021-9614(71)80108-8
	3	298.15		10.1016/S0021-9614(71)80108-8
[4 rows x 13 columns]				

We will also define the set of force field parameters which we wish to use to estimate this data set of properties. The framework has support for estimating force field parameters from a range of sources, including those in the OpenFF SMIRNOFF format, those which can be applied by AmberTools, *and more*.

Each source of a force field has a corresponding source object in the framework. In this tutorial we will be using the OpenFF Parsley force field which is based off of the SMIRNOFF format:

```
[6]: from openff.evaluator.forcefield import SmirnoffForceFieldSource
```

```
force_field_path = "openff-1.0.0.offxml"
force_field_source = SmirnoffForceFieldSource.from_path(force_field_path)
```

2.6.2 Defining the Calculation Schemas

The next step we will take will be to define a custom calculation schema for each type of property in our data set.

A calculation schema is the blueprint for how a type of property should be calculated using a particular *calculation approach*, such as directly by simulation, by reprocessing cached simulation data or, in future, a range of other options.

The framework has built-in schemas defining how densities and H_{vap} should be estimated from molecular simulation, covering all aspects from coordinate generation, force field assignment, energy minimisation, equilibration and finally the production simulation and data analysis. All of this functionality is implemented via the frameworks built-in, lightweight *workflow engine*, however we won't dive into the details of this until a later tutorial.

For the purpose of this tutorial, we will simply modify the default calculation schemas to reduce the number of molecules to include in our simulations to speed up the calculations. This step can be skipped entirely if the default options (which we recommend using for 'real-world' calculations) are to be used:

```
[7]: from openff.evaluator.properties import Density, EnthalpyOfVaporization
```

```
density_schema = Density.default_simulation_schema(n_molecules=256)
h_vap_schema = EnthalpyOfVaporization.default_simulation_schema(n_molecules=256)
```

We could further use this method to set either the absolute or the relative uncertainty that the property should be estimated to within. If either of these are set, the simulations will automatically be extended until the target uncertainty in the property has been met.

For our purposes however we won't set any targets, leaving the simulations to run for the default 1 ns.

To use these custom schemas, we need to add them to the a request options object which defines all of the options for estimating our data set:

```
[8]: from openff.evaluator.client import RequestOptions
```

Create an options object which defines how the data set should be estimated.

(continues on next page)

(continued from previous page)

```
estimation_options = RequestOptions()
# Specify that we only wish to use molecular simulation to estimate the data set.
estimation_options.calculation_layers = ["SimulationLayer"]
# Add our custom schemas, specifying that the should be used by the 'SimulationLayer'
estimation_options.add_schema("SimulationLayer", "Density", density_schema)
estimation_options.add_schema("SimulationLayer", "EnthalpyOfVaporization", h_vap_schema)
```

2.6.3 Launching the Server

The framework is split into two main applications - an EvaluatorServer and an EvaluatorClient.

The EvaluatorServer is the main object which will perform any and all calculations needed to estimate sets of properties. It is design to run on whichever compute resources you may have available (whether that be a single machine or a high performance cluster), wait until a user requests a set of properties be estimated, and then handle that request.

The EvaluatorClient is the object used by the user to send requests to estimate data sets to running server instances over a TCP connection. It is also used to query the server to see when that request has been fulfilled, and to pull back any results.

Let us begin by spawning a new server instance.

To launch a server, we need to define how this object is going to interact with the compute resource it is running on.

This is accomplished using a *calculation backend*. While there are several to choose from depending on your needs, well will go with a simple dask based one designed to run on a single machine:

```
[9]: from openff.evaluator.backends import ComputeResources
from openff.evaluator.backends.dask import DaskLocalCluster
calculation_backend = DaskLocalCluster(
    number_of_workers=1,
    resources_per_worker=ComputeResources(
        number_of_threads=1,
        number_of_gpus=1,
        preferred_gpu_toolkit=ComputeResources.GPUToolkit.CUDA
    ),
)
calculation_backend.start()
```

Here we have specified that we want to run our calculations on a single worker which has access to a single GPU.

With that defined, we can go ahead and spin up the server:

```
[10]: from openff.evaluator.server import EvaluatorServer
```

```
evaluator_server = EvaluatorServer(calculation_backend=calculation_backend)
evaluator_server.start(asynchronous=True)
```

02:47:53.961 INFO Server listening at port 8000

The server will run asynchronously in the background waiting until a client connects and requests that a data set be estimated.

2.6.4 Estimating the Data Set

With the server spun up we can go ahead and connect to it using an EvaluatorClient and request that it estimate our data set using the custom options we defined earlier:

```
[11]: from openff.evaluator.client import EvaluatorClient
evaluator_client = EvaluatorClient()
request, exception = evaluator_client.request_estimate(
    property_set=data_set,
    force_field_source=force_field_source,
    options=estimation_options,
)
assert exception is None
02:47:54.012 INFO Received estimation request from ('127.0.0.1', 50618)
```

The server will now receive the requests and begin whirring away fulfilling it. It should be noted that the request_estimate() function returns two values - a request object, and an exception object. If all went well (as it should do here) the exception object will be None.

The request object represents the request which we just sent to the server. It stores the unique id which the server assigned to the request, as well as the address of the server that the request was sent to.

The request object is primarily used to query the current state of our request, and to pull down the results when it the request finishes. Here we will use it it synchronously query the server every 30 seconds until our request has completed.

```
[12]: # Wait for the results.
```

```
results, exception = request.results(synchronous=True, polling_interval=30)
assert exception is None
```

Note: we could also asynchronously query for the results of the request. The resultant results object would then contain the partial results of any completed estimates, as well as any exceptions raised during the estimation.

2.6.5 Inspecting the Results

Now that the server has finished estimating our data set and returned the results to us, we can begin to inspect the results of the calculations:

```
[13]: print(len(results.queued_properties))
```

```
print(len(results.estimated_properties))
```

```
print(len(results.unsuccessful_properties))
print(len(results.exceptions))
```

0 4 0

0

We can (hopefully) see here that there were no exceptions raised during the calculation, and that all of our properties were successfully estimated.

We will extract the estimated data set and save this to disk:

[14]: results.estimated_properties.json("estimated_data_set.json", format=True);

2.6.6 Conclusion

And that concludes the second tutorial. In the next tutorial we will be performing some basic analysis on our estimated results.

If you have any questions and / or feedback, please open an issue on the GitHub issue tracker.

2.7 Tutorial 03 - Analysing Data Sets

In this tutorial we will be analysing the results of the calculations which we performed in the *second tutorial*. The tutorial will cover:

- comparing the estimated data set with the experimental data set.
- plotting the two data sets.

Note: If you are running this tutorial in google colab you will need to run a setup script instead of following the installation instructions:

" //iui corab_secup.rpyiib

For the sake of clarity all warnings will be disabled in this tutorial:

```
[2]: import warnings
warnings.filterwarnings('ignore')
import logging
logging.getLogger("openforcefield").setLevel(logging.ERROR)
```

2.7.1 Loading the Data Sets

We will begin by loading both the experimental data set and the estimated data set:

```
[3]: from openff.evaluator.datasets import PhysicalPropertyDataSet
experimental_data_set_path = "filtered_data_set.json"
estimated_data_set_path = "estimated_data_set.json"
# If you have not yet completed the previous tutorials or do not have the data set files
# available, copies are provided by the framework:
# from openff.evaluator.utils import get_data_filename
# experimental_data_set_path = get_data_filename(
# "tutorials/tutorial01/filtered_data_set.json"
# )
# estimated_data_set_path = get_data_filename(
(continues on next page)
```

(continued from previous page)

```
# "tutorials/tutorial02/estimated_data_set.json"
# )
```

```
experimental_data_set = PhysicalPropertyDataSet.from_json(experimental_data_set_path)
estimated_data_set = PhysicalPropertyDataSet.from_json(estimated_data_set_path)
```

if everything went well from the previous tutorials, these data sets will contain the density and H_{vap} of ethanol and isopropanol:

```
[4]: experimental_data_set.to_pandas().head()
```

```
        [4]:
        Temperature (K)
        ...
        Source

        0
        298.15
        ...
        10.1016/j.fluid.2013.10.034

        1
        298.15
        ...
        10.1021/je1013476

        2
        298.15
        ...
        10.1016/S0021-9614(71)80108-8

        3
        298.15
        ...
        10.1016/S0021-9614(71)80108-8
```

[4 rows x 13 columns]

```
[5]: estimated_data_set.to_pandas().head()
```

```
[5]: Temperature (K) ... Source
0 298.15 ... SimulationLayer
1 298.15 ... SimulationLayer
2 298.15 ... SimulationLayer
3 298.15 ... SimulationLayer
[4 rows x 13 columns]
```

2.7.2 Extracting the Results

We will now compare how the value of each property estimated by simulation deviates from the experimental measurement.

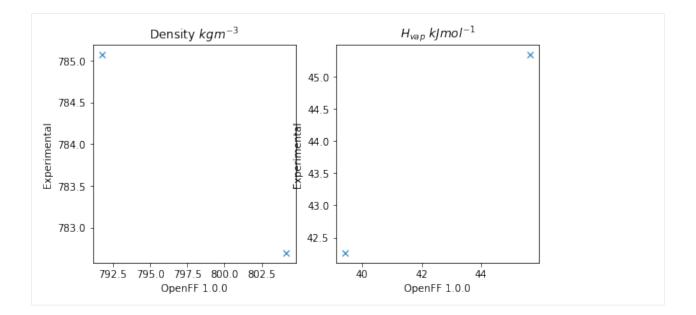
To do this we will extract a list which contains pairs of experimental and evaluated properties. We can easily match properties based on the unique ids which were automatically assigned to them on their creation:

```
[6]: properties_by_type = {
    "Density": [],
    "EnthalpyOfVaporization": []
}
for experimental_property in experimental_data_set:
    # Find the estimated property which has the same id as the
    # experimental property.
    estimated_property = next(
        x for x in estimated_data_set if x.id == experimental_property.id
    )
    # Add this pair of properties to the list of pairs
    property_type = experimental_property.__class_..__name___
    properties_by_type[property_type].append((experimental_property, estimated_property))
```

2.7.3 Plotting the Results

We will now compare the experimental results to the estimated ones by plotting them using matplotlib:

```
[7]: from matplotlib import pyplot
    # Create the figure we will plot to.
    figure, axes = pyplot.subplots(nrows=1, ncols=2, figsize=(8.0, 4.0))
    # Set the axis titles
    axes[0].set_xlabel('OpenFF 1.0.0')
    axes[0].set_ylabel('Experimental')
    axes[0].set_title('Density $kg m^{-3}$')
    axes[1].set_xlabel('OpenFF 1.0.0')
    axes[1].set_ylabel('Experimental')
    axes[1].set_title('$H_{vap}$ $kJ mol^{-1}$')
    # Define the preferred units of the properties
    from openff.evaluator import unit
    preferred_units = {
         "Density": unit.kilogram / unit.meter ** 3,
         "EnthalpyOfVaporization": unit kilojoule / unit mole
    }
    for index, property_type in enumerate(properties_by_type):
        experimental_values = []
        estimated_values = []
        preferred_unit = preferred_units[property_type]
        # Convert the values of our properties to the preferred units.
        for experimental_property, estimated_property in properties_by_type[property_type]:
            experimental_values.append(
                 experimental_property.value.to(preferred_unit).magnitude
            )
            estimated_values.append(
                 estimated_property.value.to(preferred_unit).magnitude
            )
        axes[index].plot(
            estimated_values, experimental_values, marker='x', linestyle='None'
        )
```



2.7.4 Conclusion

And that concludes the third tutorial!

If you have any questions and / or feedback, please open an issue on the GitHub issue tracker.

2.8 Tutorial 04 - Optimizing Force Fields

In this tutorial we will be using the OpenFF Evaluator framework in combination with the fantastic ForceBalance software to optimize a molecular force field against the physical property data set we created in the *first tutorial*.

ForceBalance offers a suite of tools for optimizing molecular force fields against a set of target data. Perhaps one of the most fundamental targets to fit against is experimental physical property data. Physical property data has been used extensively for decades to inform the values of non-bonded Van der Waals (VdW) interaction parameters (often referred to as Lennard-Jones parameters).

ForceBalance is seamlessly integrated with the evaluator framework, using it to evaluate the deviations between target experimentally measured data points and those evaluated using the force field being optimized (as well as the gradient of those deviations with respect to the force field parameters being optimized).

The tutorial will cover:

- setting up the input files and directory structure required by ForceBalace.
- setting up an EvaluatorServer for ForceBalance to connect to.
- running ForceBalance using those input files.
- extracting and plotting a number of statistics output during the optimization.

Note: If you are running this tutorial in google colab you will need to run a setup script instead of following the installation instructions:

For this tutorial make sure that you are using a GPU accelerated runtime.

For the sake of clarity all warnings will be disabled in this tutorial:

```
[2]: import warnings
warnings.filterwarnings('ignore')
import logging
logging.getLogger("openforcefield").setLevel(logging.ERROR)
```

We will also enable time-stamped logging to help track the progress of our calculations:

```
[3]: from openff.evaluator.utils import setup_timestamp_logging
    setup_timestamp_logging()
```

2.8.1 Setting up the ForceBalance Inputs

In this section we will be creating the directory structure required by *ForceBalance*, and populating it with the required input files.

Creating the Directory Structure

To begin with, we will create a directory to store the starting force field parameters in:

```
[4]: !mkdir forcefield
```

and one to store the input parameters for our 'fitting target' - in this case a data set of physical properties:

```
[5]: !mkdir -p targets/pure_data
```

Defining the Training Data Set

With the directories created, we will next specify the data set of physical properties which we will be training the force field against:

```
[6]: # For convenience we will use the copy shipped with the framework
from openff.evaluator.utils import get_data_filename
data_set_path = get_data_filename("tutorials/tutorial01/filtered_data_set.json")
# Load the data set.
from openff.evaluator.datasets import PhysicalPropertyDataSet
data_set = PhysicalPropertyDataSet.from_json(data_set_path)
# Due to a small bug in ForceBalance we need to zero out any uncertainties
# which are undefined. This will be fixed in future versions.
from openff.evaluator.attributes import UNDEFINED
for physical_property in data_set:
```

(continues on next page)

(continued from previous page)

```
if physical_property.uncertainty != UNDEFINED:
    continue
```

physical_property.uncertainty = 0.0 * physical_property.default_unit()

To speed up the runtime of this tutorial, we will only train the force field against measurements made for ethanol

```
[7]: data_set.filter_by_smiles("CCO")
```

in real optimizations however the data set should be much larger than two data points!

With those changes made, we can save the data set in our targets directory:

```
[8]: # Store the data set in the `pure_data` targets folder:
    data_set.json("targets/pure_data/training_set.json");
```

Defining the Starting Force Field Parameters

We will use the OpenFF Parsley 1.0.0 force field as the starting parameters for the optimization. These can be loaded directly into an OpenFF ForceField object using the OpenFF toolkit:

```
[9]: from openforcefield.typing.engines.smirnoff import ForceField
force_field = ForceField('openff-1.0.0.offxml')
```

In order to use these parameters in *ForceBalance*, we need to 'tag' the individual parameters in the force field that we wish to optimize. The toolkit easily enables us to add these tags using cosmetic attributes:

```
[10]: # Extract the smiles of all unique components in our data set.
from openforcefield.topology import Molecule, Topology
all_smiles = set(
    component.smiles
    for substance in data_set.substances
    for component in substance.components
)
for smiles in all_smiles:
    # Find those VdW parameters which would be applied to those components.
    molecule = Molecule.from_smiles(smiles)
    topology = Topology.from_molecules([molecule])
    labels = force_field.label_molecules(topology)[0]
    # Tag the exercised parameters as to be optimized.
    for parameter in labels["vdW"].values():
        parameter.add_cosmetic_attribute("parameterize", "epsilon, rmin_half")
```

Here we have made use of the toolkit's handy label_molecules function to see which VdW parameters will be assigned to the molecules in our data set, and tagged them to be parameterized.

With those tags added, we can save the parameters in the forcefield directory:

```
[11]: # Save the annotated force field file.
force_field.to_file('forcefield/openff-1.0.0-tagged.offxml')
```

Note: The force field parameters are stored in theOpenFF SMIRNOFF XML format.

Creating the Main Input File

Next, we will create the main *ForceBalance* input file. For the sake of brevity a default input file which ships with this framework will be used:

```
[12]: input_file_path = get_data_filename("tutorials/tutorial04/optimize.in")
```

```
# Copy the input file into our directory structure
import shutil
shutil.copyfile(input_file_path, "optimize.in")
```

[12]: 'optimize.in'

While there are many options that can be set within this file, the main options of interest for our purposes appear at the bottom of the file:

[13]: !tail -n 6 optimize.in

\$target
name pure_data
type Evaluator_SMIRNOFF
weight 1.0
openff.evaluator_input options.json
\$end

Here we have specified that we wish to create a new *ForceBalance* Evaluator_SMIRNOFF target called pure_data (corresponding to the name of the directory we created in the earlier step).

The main input to this target is the file path to an options.json file - it is this file which will specify all the options which should be used when *ForceBalance* requests that our target data set be estimated using the current sets of force field parameters.

We will create this file in the targets/pure_data directory later in this section.

The data set is the JSON serialized representation of the PhysicalPropertyDataSet we created during the *first tutorial*.

Defining the Estimation Options

The final step before we can start the optimization is to create the set of options which will govern how our data set is estimated using the Evaluator framework.

These options will be stored in an Evaluator_SMIRNOFF object:

```
[14]: from forcebalance.evaluator_io import Evaluator_SMIRNOFF
```

```
# Create the ForceBalance options object
target_options = Evaluator_SMIRNOFF.OptionsFile()
# Set the path to the data set
target_options.data_set_path = "training_set.json"
```

This object exposes both a set of *ForceBalance* specific options, as well as the set of Evaluator options.

The *ForceBalance* specific options allow us to define how each type of property will contribute to the optimization objective function (the value which we are trying to minimize):

$$\Delta(\theta) = \sum_{n}^{N} \frac{weight_n}{M_n} \sum_{m}^{M_n} \left(\frac{y_m^{ref} - y_m(\theta)}{denominator_n} \right)^2$$

where N is the number of types of properties (e.g. density, enthalpy of vaporization, etc.), M_n is the number of data points of type n, y_m^{ref} is the experimental value of data point m and $y_m(\theta)$ is the estimated value of data point m using the current force field parameters

In particular, the options object allows us to specify both an amount to scale each type of properties contribution to the objective function by $(weight_n)$, and the amount to scale the difference between the experimental and estimated properties $(denominator_n)$:

```
[15]: from openff.evaluator import unit
```

```
target_options.weights = {
    "Density": 1.0,
    "EnthalpyOfVaporization": 1.0
}
target_options.denominators = {
    "Density": 30.0 * unit.kilogram / unit.meter ** 3,
    "EnthalpyOfVaporization": 3.0 * unit.kilojoule / unit.mole
}
```

where here we have chosen values that ensure that both types of properties contribute roughly equally to the total objective function.

The Evaluator specific options correspond to a standard RequestOptions object:

```
[16]: from openff.evaluator.client import RequestOptions
```

```
# Create the options which evaluator should use.
evaluator_options = RequestOptions()
# Choose which calculation layers to make available.
evaluator_options.calculation_layers = ["SimulationLayer"]
# Reduce the default number of molecules
from evaluator.properties import Density, EnthalpyOfVaporization
density_schema = Density.default_simulation_schema(n_molecules=256)
h_vap_schema = EnthalpyOfVaporization.default_simulation_schema(n_molecules=256)
evaluator_options.add_schema("SimulationLayer", "Density", density_schema)
evaluator_options.add_schema("SimulationLayer", "EnthalpyOfVaporization", h_vap_schema)
target_options.estimation_options = evaluator_options
```

These options allow us to control exactly how each type of property should be estimated, which calculation approaches should be used and more. Here we use the same options are were used in the *second tutorial*

Note: more information about the different estimation options can befound here

And that's the options created! We will finish off by serializing the options into our target directory:

```
[17]: # Save the options to file.
with open("targets/pure_data/options.json", "w") as file:
    file.write(target_options.to_json())
```

2.8.2 Launching an Evaluator Server

With the *ForceBalance* options created, we can now move onto launching the EvaluatorServer which *ForceBalance* will call out to when it needs the data set to be evaluated:

```
[18]: # Launch the calculation backend which will distribute any calculations.
     from openff.evaluator.backends import ComputeResources
     from openff.evaluator.backends.dask import DaskLocalCluster
     calculation_backend = DaskLocalCluster(
         number_of_workers=1,
         resources_per_worker=ComputeResources(
             number_of_threads=1,
             number_of_gpus=1,
             preferred_gpu_toolkit=ComputeResources.GPUToolkit.CUDA
         ),
     )
     calculation_backend.start()
      # Launch the server object which will listen for estimation requests and schedule any
     # required calculations.
     from openff.evaluator.server import EvaluatorServer
     evaluator_server = EvaluatorServer(calculation_backend=calculation_backend)
     evaluator_server.start(asynchronous=True)
     01:30:20.505 INFO
                            Server listening at port 8000
```

We will not go into the details of this here as this was already covered in the second tutorial

2.8.3 Running ForceBalance

With the inputs created and an Evaluator server spun up, we are finally ready to run the optimization! This can be accomplished with a single command:

```
[19]: !ForceBalance optimize.in
```

If everything went well *ForceBalance* should exit cleanly, and will have stored out newly optimized force field in the results directory.

```
[20]: !ls result/optimize
```

```
openff-1.0.0-tagged_1.offxml openff-1.0.0-tagged.offxml
```

2.8.4 Plotting the results

As a last step in this tutorial, we will extract the objective function at each iteration from the *ForceBalance* output files and plot this using matplotlib.

First, we will extract the objective function from the pickle serialized output files which can be found in the optimize.tmp/pure_data/iter_****/ directories:

```
[21]: from forcebalance.nifty import lp_load
```

```
# Determine how many iterations ForceBalance has completed.
from glob import glob
n_iterations = len(glob("optimize.tmp/pure_data/iter*"))
```

```
# Extract the objective function at each iteration.
objective_function = []
```

for iteration in range(n_iterations):

```
folder_name = "iter_" + str(iteration).zfill(4)
file_path = f"optimize.tmp/pure_data/{folder_name}/objective.p"
```

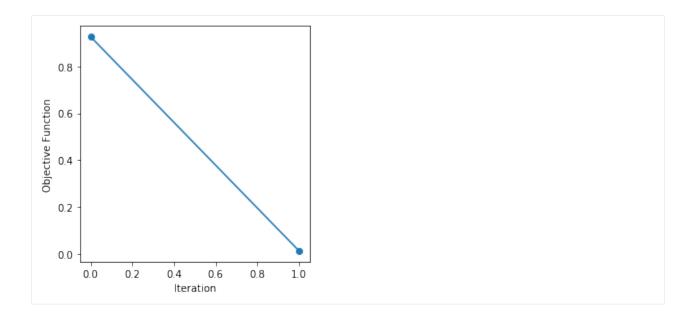
statistics = lp_load(file_path)
objective_function.append(statistics["X"])

print(objective_function)

```
[0.9270359101845124, 0.011497456194198362]
```

The objective function is then easily plotted:

```
[22]: from matplotlib import pyplot
figure, axis = pyplot.subplots(1, 1, figsize=(4, 4))
axis.set_xlabel("Iteration")
axis.set_ylabel("Objective Function")
axis.plot(range(n_iterations), objective_function, marker="o")
figure.tight_layout()
```



2.8.5 Conclusion

And that concludes the fourth tutorial!

If you have any questions and / or feedback, please open an issue on the GitHub issue tracker.

2.9 Property Data Sets

A *PhysicalPropertyDataSet* is a collection of measured physical properties encapsulated as *physical property* objects. They may be created from scratch:

```
# Define a density measurement
density = Density(
    substance=Substance.from_components("0"),
    thermodynamic_state=ThermodynamicState(
        pressure=1.0*unit.atmospheres, temperature=298.15*unit.kelvin
    ),
    phase=PropertyPhase.Liquid,
    value=1.0*unit.gram/unit.millilitre,
    uncertainty=0.0001*unit.gram/unit.millilitre
)
# Add the property to a data set
data_set = PhysicalPropertyDataset()
data_set.add_properties(density)
```

are readily JSON (de)serializable:

```
# Save the data set as a JSON file.
data_set.json(file_path="data_set.json", format=True)
# Load the data set from a JSON file
data_set = PhysicalPropertyDataset.from_json(file_path="data_set.json")
```

and may be converted to pandas DataFrame objects:

data_set.to_pandas()

The framework implements specific data set objects for extracting data measurements directly from a number of open data sources, such as the *ThermoMLDataSet* (see *ThermoMLArchive*) which provides utilities for extracting the data from the NIST ThermoML Archive and converting it into the standard framework objects.

Data set objects are directly iterable:

for physical_property in data_set:

or can be iterated over for a specific substance:

```
for physical_property in data_set.properties_by_substance(substance):
```

...

or for a specific type of property:

```
for physical_property in data_set.properties_by_type("Density"):
```

. . .

2.9.1 Physical Properties

The *PhysicalProperty* object is a base class for any object which describes a measured property of substance, and is defined by a combination of:

- the observed value of the property.
- Substance specifying the substance that the measurement was collected for.
- PropertyPhase specifying the phase that the measurement was collected in.
- ThermodynamicState specifying the thermodynamic conditions under which the measurement was performed

as well as optionally

- the uncertainty in the value of the property.
- a list of *ParameterGradient* which defines the gradient of the property with respect to the model parameters if it was computationally estimated.
- a Source specifying the source (either experimental or computational) and provenance of the measurement.

Each type of property supported by the framework, such as a density of an enthalpy of vaporization, must have it's own class representation which inherits from *PhysicalProperty*:

```
# Define a density measurement
density = Density(
    substance=Substance.from_components("0"),
    thermodynamic_state=ThermodynamicState(
        pressure=1.0*unit.atmospheres, temperature=298.15*unit.kelvin
    ),
    phase=PropertyPhase.Liquid,
    value=1.0*unit.gram/unit.millilitre,
    uncertainty=0.0001*unit.gram/unit.millilitre
)
```

2.9.2 Substances

A *Substance* is defined by a number of components (which may have specific roles assigned to them such as being solutes in the system) and the amount of each component in the substance.

To create a pure substance containing only water:

```
water_substance = Substance.from_components("0")
```

To create binary mixture of water and methanol in a 20:80 ratio:

```
binary_mixture = Substance()
binary_mixture.add_component(Component(smiles="0"), MoleFraction(value=0.2))
binary_mixture.add_component(Component(smiles="C0"), MoleFraction(value=0.8))
```

To create a substance of an infinitely dilute paracetamol solute dissolved in water:

```
solution = Substance()
solution.add_component(
    Component(smiles="0", role=Component.Role.Solvent), MoleFraction(value=1.0)
)
solution.add_component(
    Component(smiles="CC(=0)Nc1ccc(0)cc1", role=Component.Role.Solute),__
--ExactAmount(value=1)
)
```

2.9.3 Property Phases

The PropertyPhase enum describes the possible phases which a measurement was performed in.

While the enum only has three defined phases (Solid, Liquid and Gas), multiple phases can be formed by OR'ing (|) multiple phases together. As an example, to define a phase for a liquid and gas coexisting:

```
liquid_gas_phase = PropertyPhase.Liquid | PropertyPhase.Gas
```

2.9.4 Thermodynamic States

A *ThermodynamicState* specifies a combination of the temperature and (optionally) the pressure at which a measurement is performed:

```
thermodynamic_state = ThermodynamicState(
    temperature=298.15*unit.kelvin, pressure=1.0*unit.atmosphere
)
```

2.10 ThermoML Archive

The *ThermoMLDataSet* object offers an API for extracting physical properties from the NIST ThermoML Archive, both directly from the archive itself or from files stored in the IUPAC- standard ThermoML format.

The API only supports extracting those properties which have been *registered* with the frameworks plug-in system, and does not currently load the full set of metadata available in the archive files.

Note: If the metadata you require is not currently exposed, please open an issue on the GitHub issue tracker to request it.

Currently the framework has built-in support for extracting:

- Mass density, kg/m3 (Density)
- Excess molar volume, m3/mol (ExcessMolarVolume)
- Relative permittivity at zero frequency (DielectricConstant)
- Excess molar enthalpy (molar enthalpy of mixing), kJ/mol (EnthalpyOfMixing)
- Molar enthalpy of vaporization or sublimation, kJ/mol (EnthalpyOfVaporization)

where here both the ThermoML property name (as defined by the IUPAC XML schema) and the built-in framework class are listed.

2.10.1 Registering Properties

Properties to be extracted from ThermoML archives must have a corresponding class representation to be loading into. This class representation must both:

- inherit from the frameworks PhysicalProperty class and
- be registered with the frameworks plug-in system using either the thermoml_property() decorator or the register_thermoml_property() method.

As an example, a class representation of the ThermoML '*Mass density*, *kg/m3*' property could be defined and registered with the plug-in system using:

```
@thermoml_property("Mass density, kg/m3", supported_phases=PropertyPhase.Liquid)
class Density(PhysicalProperty):
    """A class representation of a mass density property"""
```

The thermoml_property() decorator takes in the name of the ThermoML property (as defined by the IUPAC schema) as well as the phases which the framework will be able to estimate this property in.

Multiple ThermoML properties can be mapped onto a single class using the flexible *register_thermoml_property()* function. For example, the '*Specific volume, m3/kg*' property (which is simply the reciprocal of mass density) may be mapped onto the *Density* object by providing a conversion_function:

```
def specific_volume_to_mass_density(specific_volume):
    """Converts a specific volume measurement into a mass
    density.
    Parameters
    ------
```

```
specific_volume: ThermoMLProperty
        The specific volume measurement to convert.
    .....
   mass_density = Density()
   mass_density.value = 1.0 / specific_volume.value
   if mass_density.uncertainty is not None:
        mass_density.uncertainty = 1.0 / mass_density.uncertainty
   mass_density.phase = specific_volume.phase
   mass_density.thermodynamic_state = specific_volume.thermodynamic_state
   mass_density.substance = specific_volume.substance
   return mass_density
# Register the ThermoML property using the conversion function.
register_thermoml_property(
    thermoml_string="Specific volume, m3/kg",
    supported_phases=PropertyPhase.Liquid,
   property_class=Density,
    conversion_function=specific_volume_to_mass_density
)
```

Converting the different density derivatives into a single density class removes the need to produce many very similar class representations of density measurements, and allows a single calculation schema to be defined for all variants.

2.10.2 Loading Data Sets

Data sets are most easily loaded using their digital object identifiers (DOI). For example, to retrieve the ThermoML data set that accompanies this paper, we can simply use the DOI 10.1016/j.jct.2005.03.012:

```
data_set = ThermoMLDataset.from_doi('10.1016/j.jct.2005.03.012')
```

Data can be pulled from multiple sources at once by specifying multiple identifiers:

```
identifiers = ['10.1021/acs.jced.5b00365', '10.1021/acs.jced.5b00474']
dataset = ThermoMLDataset.from_doi(*identifiers)
```

Entire archives of properties can be downloaded directly from the ThermoML website and parsed by the framework. For example, to create a data set object containing all of the measurements recorded from the International Journal of Thermophysics:

```
# Download the archive of all properties from the IJT journal.
import requests
request = requests.get("https://trc.nist.gov/ThermoML/IJT.tgz", stream=True)
# Make sure the request went ok.
assert request
# Unzip the files into a new 'ijt_files' directory.
```

```
import io, tarfile
tar_file = tarfile.open(fileobj=io.BytesIO(request.content))
tar_file.extractall("ijt_files")
# Get the names of the extracted files
import glob
file_names = glob.glob("ijt_files/*.xml")
# Create the data set object
from openff.evaluator.datasets.thermoml import ThermoMLDataSet
data_set = ThermoMLDataSet.from_file(*file_names)
# Save the data set to a JSON object
data_set.json(file_path="ijt.json", format=True)
```

2.11 Taproom

The *TaproomDataSet* object offers an API for retrieving host-guest binding affinity measurements from the curated taproom repository.

Note: taproom may be installed by running conda install -c conda-forge taproom

This includes retrieving all of the data available:

```
from openff.evaluator.datasets.taproom import TaproomDataSet
taproom_set = TaproomDataSet()
```

data measure for a single host molecule (e.g. alpha-cyclodextrin):

```
acd_taproom_set = TaproomDataSet(host_codes=["acd"])
```

or data for a particular host and guest pair:

acd_taproom_set = TaproomDataSet(host_codes=["acd"], guest_codes=["bam"])

All measurements in this data set have an associated *TaproomSource* as their source provenance. This tracks both the original source of the measurement as well as the taproom identifier.

Note: Currently the data set object will assume a default set of buffer conditions (either no buffer, or a buffer of a salt with a specified ionic strength) rather than reading the buffer from the taproom measurement directory. This is consistent with previous applications of the data set.

2.12 Data Set Curation

The framework offers a full suite of features to facilitate the curation of data sets of physical properties, including:

• a significant amount of data filters, including to filter by state, substance composition and chemical functionalities.

and components to

- easily download and import the full NIST ThermoML and FreeSolv archives .
- select data points which were measured close to a set of target states, and which were measured for a diverse range of substances which contain specific functionalities.
- convert between different compatible property types (e.g. convert density <-> excess molar volume data).

These features are implemented as *CurationComponent* objects, which take as input an associated *CurationComponentSchema* which controls how the curation components should be applied to a particular data set (or a data set which is being stored as pandas DataFrame object).

An example of a curation component would be one that filters out data points which were measured outside of a particular temperature range:

```
# Filter data points measured at less than 290.0 K or greater than 320.0 K
filtered_frame = FilterByTemperature.apply(
    data_frame,
    FilterByTemperatureSchema(minimum_temperature=290.0, maximum_temperature=320.0),
)
```

Curation components can be conveniently chained together using a *CurationWorkflow* and an associated *CurationWorkflowSchema* so as to easily curated full training and testing data sets:

```
curation_schema = WorkflowSchema(
   component_schemas=[
        # Import the ThermoML archive.
        thermoml.ImportThermoMLDataSchema()
        # Filter out any measurements made for systems with more than two components
        filtering.FilterByNComponentsSchema(n_components=[1, 2]),
        # Remove any duplicate data.
        filtering.FilterDuplicatesSchema(),
        # Filter out data points measured away from ambient
        # and biologically relevant temperatures.
        filtering.FilterByTemperatureSchema(
            minimum_temperature=298.0, maximum_temperature=320.0
       ),
        # Retain only density and enthalpy of mixing data points.
        filtering.FilterByPropertyTypesSchema(
            property_types=["Density", "EnthalpyOfMixing"],
       ),
        # Select data points measured for alcohols, esters or mixtures of both.
        selection.SelectSubstancesSchema(
            target_environments=[
                ChemicalEnvironment.Alcohol,
                ChemicalEnvironment.CarboxylicAcidEster,
            ],
            n_per_environment=10,
```

```
),
]
)
data_frame = Workflow.apply(pandas.DataFrame(), curation)
```

2.12.1 Examples

Data Extraction

• *ImportFreeSolv*: A component which will download the latest, full FreeSolv data set from the GitHub repository:

```
from openff.evaluator.datasets.curation.components.freesolv import (
    ImportFreeSolv,
    ImportFreeSolvSchema,
)
# Import the full FreeSolv data set.
data_frame = ImportFreeSolv.apply(pandas.DataFrame(), ImportFreeSolvSchema())
```

• *ImportThermoMLData*: A component which will download all *supported data* from the NIST ThermoML Archive:

```
from openff.evaluator.datasets.curation.components.thermoml import (
    ImportThermoMLData,
    ImportThermoMLDataSchema,
)
# Import all data collected from the IJT journal.
data_frame = ImportThermoMLData.apply(
    pandas.DataFrame(), ImportThermoMLDataSchema(journal_names=["IJT"])
)
```

Filtration

• *FilterDuplicates*: A component to remove duplicate data points (within a specified precision) from a data set:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterDuplicates,
    FilterDuplicatesSchema,
)
filtered_frame = FilterDuplicates.apply(data_frame, FilterDuplicatesSchema())
```

• *FilterByTemperature*: A component which will filter out data points which were measured outside of a specified temperature range:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterByTemperature,
    FilterByTemperatureSchema,
)
filtered_frame = FilterByTemperature.apply(
    data_frame,
    FilterByTemperatureSchema(minimum_temperature=290.0, maximum_temperature=320.0),
)
```

• *FilterByPressure*: A component which will filter out data points which were measured outside of a specified pressure range:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterByPressure,
    FilterByPressureSchema,
)
filtered_frame = FilterByPressure.apply(
    data_frame,
    FilterByPressureSchema(minimum_pressure=100.0, maximum_pressure=140.0),
)
```

• *FilterByMoleFraction*: A component which will filter out data points which were measured outside of a specified mole fraction range:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterByMoleFraction,
    FilterByMoleFractionSchema,
)
filtered_frame = FilterByMoleFraction.apply(
    data_frame, FilterByMoleFractionSchema(mole_fraction_ranges={2: [[(0.1, 0.3)]]})
)
```

• FilterByRacemic: A component which will filter out data points which were measured for racemic mixtures:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterByRacemic,
    FilterByRacemicSchema,
)
filtered_frame = FilterByRacemic.apply(data_frame, FilterByRacemicSchema())
```

• *FilterByElements*: A component which will filter out data points which were measured for systems which contain specific elements:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterByElements,
    FilterByElementsSchema,
)
filtered_frame = FilterByElements.apply(
```

```
data_frame,
FilterByElementsSchema(allowed_elements=["C", "0", "H"]),
```

• *FilterByPropertyTypes*: A component which will apply a filter which only retains properties of specified types:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterByPropertyTypes,
    FilterByPropertyTypesSchema,
)

# Retain only density measurements made for either pure or binary systems.
filtered_frame = FilterByPropertyTypes.apply(
    data_frame,
    FilterByPropertyTypesSchema(
        property_types=["Density"],
        n_components={"Density": [1, 2]},
    ),
)
```

• *FilterByStereochemistry*: A component which filters out data points measured for systems whereby the stereochemistry of a number of components is undefined:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterByStereochemistry,
    FilterByStereochemistrySchema,
)
filtered_frame = FilterByStereochemistry.apply(
    data_frame, FilterByStereochemistrySchema()
)
```

• *FilterByCharged*: A component which filters out data points measured for substance where any of the constituent components have a net non-zero charge.:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterByCharged,
    FilterByChargedSchema,
)
```

filtered_frame = FilterByCharged.apply(data_frame, FilterByChargedSchema())

 FilterByIonicLiquid: A component which filters out data points measured for substances which contain or are classed as an ionic liquids:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterByIonicLiquid,
    FilterByIonicLiquidSchema,
)
```

filtered_frame = FilterByIonicLiquid.apply(data_frame, FilterByIonicLiquidSchema())

)

• *FilterBySmiles*: A component which filters the data set so that it only contains either a specific set of smiles, or does not contain any of a set of specifically excluded smiles:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterBySmiles,
    FilterBySmilesSchema,
)
filtered_frame = FilterBySmiles.apply(
    data_frame, FilterBySmilesSchema(smiles_to_include=["CCCO"]),
)
```

• *FilterBySmirks*: A component which filters a data set so that it only contains measurements made for molecules which contain (or don't) a set of chemical environments represented by SMIRKS patterns:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterBySmirks,
    FilterBySmirksSchema,
)
filtered_frame = FilterBySmirks.apply(
    data_frame, FilterBySmirksSchema(smirks_to_include=["[#6a]"]),
)
```

 FilterByNComponents: A component which filters out data points measured for systems with specified number of components:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterByNComponents,
    FilterByNComponentsSchema,
)
filtered_frame = FilterByNComponents.apply(
    data_frame, FilterByNComponentsSchema(n_components=[1, 2])
)
```

• *FilterBySubstances*: A component which filters the data set so that it only contains properties measured for particular substances:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterBySubstances,
    FilterBySubstancesSchema,
)
filtered_frame = FilterBySubstances.apply(
    data_frame, FilterBySubstancesSchema(substances_to_include=[("CO", "C")])
)
```

• *FilterByEnvironments*: A component which filters a data set so that it only contains measurements made for substances which contain specific chemical environments:

```
from openff.evaluator.datasets.curation.components.filtering import (
    FilterByEnvironments,
    FilterByEnvironmentsSchema,
```

```
)
filtered_frame = FilterByEnvironments.apply(
    data_frame,
    FilterByEnvironmentsSchema(
        environments=[
            ChemicalEnvironment.Aqueous,
            ChemicalEnvironment.Alcohol,
            ChemicalEnvironment.Amine,
        ]
     ),
)
```

Data Selection

• SelectSubstances: A component for selecting data points which were measured for specified number of maximally diverse systems containing a specified set of chemical functionalities:

```
# Select (if possible) data points which were measured for 10 different (and
# structurally diverse) alcohols.
schema = SelectSubstancesSchema(
    target_environments=[ChemicalEnvironment.Alcohol],
    n_per_environment=10,
)
data_frame = ConvertExcessDensityData.apply(data_frame, schema)
```

• SelectDataPoints: A component for selecting a set of data points which are close to a particular set of states:

Data Conversion

• *ConvertExcessDensityData*: A component for converting binary mass density data to excess molar volume data and vice versa where pure density data measured for the components is available:

```
from openff.evaluator.datasets.curation.components.conversion import (
        ConvertExcessDensityData,
        ConvertExcessDensityDataSchema,
)
converted_data_frame = ConvertExcessDensityData.apply(
        data_frame, ConvertExcessDensityDataSchema()
)
```

2.13 Physical Properties

A core philosophy of this framework is that users should be able to seamlessly curate data sets of physical properties and then estimate that data set using computational methods without significant user intervention and using sensible, well validated workflows.

This page aims to provide an overview of which physical properties are supported by the framework and how they are computed using the different *calculation layers*.

In this document $\langle X \rangle$ will be used to denote the ensemble average of an observable X.

2.13.1 Density

The density (ρ) is computed according to

$$\rho = \left\langle \frac{M}{V} \right\rangle$$

where M and V are the total molar mass and volume the system respectively.

Direct Simulation

The density is estimated using the default *simulation workflow* without modification. The estimation of liquid densities is assumed.

MBAR Reweighting

The density is estimated using the default *reweighting workflow* without modification. The estimation of liquid densities is assumed.

2.13.2 Dielectric Constant

The dielectric constant (ε) is computed from the fluctuations in a systems dipole moment (see Equation 7 of [1]) according to:

$$\varepsilon = 1 + \frac{\left\langle \vec{\mu}^2 \right\rangle - \left\langle \vec{\mu} \right\rangle^2}{3\varepsilon_0 \left\langle V \right\rangle k_b T}$$

where $\vec{\mu}$, V are the systems dipole moment and volume respectively, k_b the Boltzmann constant, T the temperature, and ε_0 the permittivity of free space.

Note: In *v0.2.2* and earlier of the framework the variance was computed as $\langle (\vec{\mu} - \langle \vec{\mu} \rangle)^2 \rangle$ in order to match the mdtraj implementation which has been used in previous studies by the OpenFF Consortium (see for example [2]). The two approaches should be numerically indistinguishable however.

Direct Simulation

The dielectric is estimated using the default *simulation workflow* which has been modified to use the specialized *AverageDielectricConstant* protocol in place of the default *AverageObservable* protocol. The estimation of liquid dielectric constants is assumed.

MBAR Reweighting

The dielectric is estimated using the default *reweighting workflow* which has been modified to use the specialized *ReweightDielectricConstant* protocol in place of the default *ReweightObservable* protocol. It should be noted that the *ReweightDielectricConstant* protocol employs bootstrapping to compute the uncertainty in the average dielectric constant, rather than attempting to propagate uncertainties in the average dipole moments and volumes. The estimation of liquid dielectric constants is assumed.

2.13.3 Enthalpy of Vaporization

The enthalpy of vaporization ΔH_{vap} (see [3]) can be computed according to

$$\Delta H_{vap} = \langle H_{gas} \rangle - \langle H_{liquid} \rangle = \langle E_{gas} \rangle - \langle E_{liquid} \rangle + p \left(\langle V_{gas} \rangle - \langle V_{liquid} \rangle \right)$$

where H, E, and V are the enthalpy, total energy and volume respectively.

Under the assumption that $V_{gas} >> V_{liquid}$ and that the gas is ideal the above expression can be simplified to

$$\Delta H_{vap} = \langle U_{gas} \rangle - \langle U_{liquid} \rangle + RT$$

where U is the potential energy, T the temperature and R the universal gas constant. This simplified expression is computed by default by this framework.

Direct Simulation

- Liquid phase: The potential energy of the liquid phase is estimated using the default *simulation workflow*, and divided by the number of molecules in the simulation box using the divisor input of the *AverageObservable* protocol.
- Gas phase: The potential energy of the gas phase is estimated using the default *simulation workflow*, which has been modified so that
 - the simulation box only contains a single molecule.
 - all periodic boundary conditions have been disabled.
 - all simulations are performed in the NVT ensemble.
 - the production simulation is run for 15000000 steps at a time (rather than 1000000 steps).
 - all simulations are run using the OpenMM reference platform (CPU only) regardless of whether a GPU is available. This is fastest platform to use when simulating a single molecule in vacuum with OpenMM.

The final enthalpy is then computed by subtracting the gas potential energy from the liquid potential energy (*SubtractValues*) and adding the *RT* term (*AddValues*). Uncertainties are propagated through the subtraction by the normal means using the uncertainties package.

MBAR Reweighting

- Liquid phase: The potential energy of the liquid phase is estimated using the default *reweighting workflow*, and divided by the number of molecules in the simulation box using an extra *DivideValue* protocol.
- **Gas phase**: The potential energy of the gas phase is estimated using the default *reweighting workflow*, which has been modified so that all periodic boundary conditions have been disabled.

The final enthalpy is then computed by subtracting the gas potential energy from the liquid potential energy (*SubtractValues*) and adding the *RT* term (*AddValues*). Uncertainties are propagated through the subtraction by the normal means using the uncertainties package.

2.13.4 Enthalpy of Mixing

The enthalpy of mixing $\Delta H_{mix}(x_0, \cdots, x_{M-1})$ for a system of M components is computed according to

$$\Delta H_{mix}\left(x_{0},\cdots,x_{M-1}\right) = \frac{\langle H_{mix}\rangle}{N_{mix}} - \sum_{i}^{M} x_{i} \frac{\langle H_{i}\rangle}{N_{i}}$$

where H_{mix} is the enthalpy of the full mixture, and H_i , x_i are the enthalpy and the mole fraction of component *i* respectively. N_{mix} and N_i are the total number of molecules used in the full mixture simulations and the simulations of each individual component respectively.

When re-weighting cached data to compute H_{mix} we make the approximation that the kinetic energy contributions cancel out between the mixture and each of the components, and hence can be computed by only re-weighting the NPT reduced potential:

$$\Delta H_{mix}\left(x_{0},\cdots,x_{M-1}\right) \approx \frac{1}{\beta} \left(\frac{\langle u_{mix} \rangle}{N_{mix}} - \sum_{i}^{M} x_{i} \frac{\langle u_{i} \rangle}{N_{i}}\right)$$

where $u \equiv \beta (U + pV)$ is the NPT reduced potential, U the potential energy, p the pressure and V the volume.

Direct Simulation

- **Mixture**: The enthalpy of the full mixture is estimated using the default *simulation workflow* and divided by the number of molecules in the simulation box using the divisor input of the *AverageObservable* protocol.
- **Components**: The enthalpy of each of the components is estimated using the default *simulation workflow*, divided by the number of molecules in the simulation box using the divisor input of the *AverageObservable* protocol, and weighted by their mole fraction *in the mixture simulation box* using the *WeightByMoleFraction* protocol.

The final enthalpy is then computed by summing the component enthalpies (*AddValues*) and subtracting these from the mixture enthalpy (*SubtractValues*). Uncertainties are propagated through the summation and subtraction by the normal means using the uncertainties package.

MBAR Reweighting

- **Mixture**: The reduced potential of the full mixture is estimated using the default *reweighting workflow* and divided by the number of molecules in the reweighting box using an extra *DivideValue* protocol.
- **Components**: The reduced potential of each of the components is estimated using the default *reweighting work-flow*, divided by the number of molecules in the reweighting box using an extra *DivideValue* protocol, and weighted by their mole fraction using the *WeightByMoleFraction* protocol.

The final enthalpy is then computed by summing the component enthalpies (AddValues), subtracting these from the mixture enthalpy (SubtractValues), and multiplying by $1/\beta$ (MultiplyValue). Uncertainties are propagated by the normal means using the uncertainties package.

2.13.5 Excess Molar Volume

The excess molar volume $\Delta V_{excess}(x_0, \dots, x_{M-1})$ for a system of M components is computed according to

$$\Delta V_{excess}\left(x_{0},\cdots,x_{M-1}\right) = N_{A}\left(\frac{\langle V_{mix}\rangle}{N_{mix}} - \sum_{i}^{M} x_{i}\frac{\langle V_{i}\rangle}{N_{i}}\right)$$

where V_{mix} is the volume of the full mixture, and V_i , x_i are the volume and the mole fraction of component *i* respectively. N_{mix} and N_i are the total number of molecules used in the full mixture simulations and the simulations of each individual component respectively, and N_A is the Avogadro constant.

Direct Simulation

- **Mixture**: The molar volume of the full mixture is estimated using the default *simulation workflow* and divided by the molar number of molecules in the simulation box using the divisor input of the *AverageObservable* protocol.
- **Components**: The molar volume of each of the components is estimated using the default *simulation work-flow*, divided by the molar number of molecules in the simulation box using the divisor input of the *AverageObservable* protocol, and weighted by their mole fraction *in the mixture simulation box* using the *WeightByMoleFraction* protocol.

The final excess molar volume is then computed by summing the component molar volumes (*AddValues*) and subtracting these from the mixture molar volume (*SubtractValues*). Uncertainties are propagated through the summation and subtraction by the normal means using the uncertainties package.

MBAR Reweighting

- **Mixture**: The enthalpy of the full mixture is estimated using the default *reweighting workflow* and divided by the molar number of molecules in the reweighting box using an extra *DivideValue* protocol.
- **Components**: The enthalpy of each of the components is estimated using the default *reweighting workflow*, divided by the molar number of molecules in the reweighting box using an extra *DivideValue* protocol, and weighted by their mole fraction using the *WeightByMoleFraction* protocol.

The final enthalpy is then computed by summing the component enthalpies (*AddValues*) and subtracting these from the mixture enthalpy (*SubtractValues*). Uncertainties are propagated through the summation and subtraction by the normal means using the uncertainties package.

2.13.6 Solvation Free Energies

Solvation free energies are currently computed using the Yank free energy package using direct molecular simulations. By default the calculations attempt to use 2000 solvent molecules, and the alchemical lambda spacings are selected using the built-in 'trailblazing' algorithm.

See the Yank documentation for more details.

2.13.7 Host-Guest Binding Free Energy

Warning: The computation of this property is still in beta. Users are heavily recommended to validate any calculations involving this property.

Host-guest binding free energies are currently computed using the attach-pull-release (APR) method [4] through integration with the pAPRika framework.

2.14 Common Workflows

As may be expected, most of the workflows used to estimate the physical properties within the framework make use of very similar workflows. This page aims to document the built-in 'template' workflows from which the more complex physical property estimation workflows are constructed.

2.14.1 Direct Simulation

Properties being estimated using the *direct simulation* calculation layer typically base their workflows off of the *generate_simulation_protocols()* template.

Note: This template currently assumes that a liquid phase property is being computed.

The workflow produced by this template proceeds as follows:

1) 1000 molecules are inserted into a simulation box with an approximate density of 0.95 g / mL using packmol (*BuildCoordinatesPackmol*).

- the system is parameterized using either the OpenFF toolkit, TLeap or LigParGen depending on the force field being employed (BuildSmirnoffSystem, BuildTLeapSystem or BuildLigParGenSystem).
- 3) an energy minimization is performed using the default OpenMM energy minimizer (*OpenMMEnergyMinimisation*).
- 4) the system is equilibrated by running a short NPT simulation for 100000 steps using a timestep of 2 fs and using the OpenMM simulation engine (*OpenMMSimulation*).
- 5) while the uncertainty in the average observable is greater than the requested tolerance (if specified):

5a) a longer NPT production simulation is run for 1000000 steps with a timestep of 2 fs and using the OpenMM simulation protocol (*OpenMMSimulation*) with its default Langevin integrator and Monte Carlo barostat.

5b) the correlated samples are removed from the simulation outputs and the average value of the observable of interest and its uncertainty are computed by bootstrapping with replacement for 250 iterations (*AverageObservable*). See [1] for details of the decorrelation procedure.

5c) steps 5a) and 5b) are repeated until the uncertainty condition (if applicable) is met.

The decorrelated simulation outputs are then made available ready to be cached by a *storage backend* (*DecorrelateObservables*, *DecorrelateTrajectory*).

2.14.2 MBAR Reweighting

Properties being estimated using the *MBAR reweighting* calculation layer typically base their workflows off of the *generate_reweighting_protocols()* template.

The workflow produced by this template proceeds as follows:

1) for each stored simulation data:

1a) the cached data is retrieved from disk (UnpackStoredSimulationData)

- the cached data from is concatenated together to form a single trajectory of configurations and observables (ConcatenateTrajectories, ConcatenateStatistics).
- 3) for each stored simulation data:

3a) the system is parameterized using the force field parameters which were used when originally generating the cached data i.e. one of the reference states (*BuildSmirnoffSystem*, *BuildTLeapSystem* or *BuildLigParGenSystem*).

3b) the reduced potential of each configuration in the concatenated trajectory is evaluated using the parameterized system (*OpenMMEvaluateEnergies*).

4) the system is parameterized using the force field parameters with which the property of interest should be calculated using i.e. of the target state (*BuildSmirnoffSystem*, *BuildTLeapSystem* or *BuildLigParGenSystem*) and the reduced potential of each configuration in the concatenated trajectory is evaluated using the parameterized system (*OpenMMEvaluateEnergies*).

4a) (*optional*) if the observable of interest is a function of the force field parameters it is recomputed using the target state parameters. These recomputed values then replace the original concatenated observables loaded from the cached data.

5) the reference potentials, target potentials and the joined observables are sub-sampled to only retain equilibrated, uncorrelated samples (*AverageObservable*, *DecorrelateObservables*, *DecorrelateTrajectory*). See [1] for details of the decorrelation procedure.

6) the MBAR method is employed to compute the average value of the observable of interest and its uncertainty at the target state, taking the reference state reduced potentials as input. See [2] for the theory behind this approach. An exception is raised if there are not enough effective samples to reweight (*ReweightObservable*).

In more specialised cases the *generate_base_reweighting_protocols()* template (which *generate_reweighting_protocols()* is built off of) is instead used due to its greater flexibility.

2.14.3 References

2.15 Gradients

A most fundamental feature of this framework is its ability to rapidly compute the gradients of physical properties with respect to the force field parameters used to estimate them.

Note: Prior to v0.3.0 of this framework a combination of re-weighting and the central finite difference was employed to estimate the gradients of observables. From v0.3.0 onwards the fluctuation method [1] is instead used. The change was made to, in future, enable better integration with automatic differentiation libraries such as jax, and differentiable simulation engines such as timemachine.

2.15.1 Theory

The framework currently employs the fluctuation approach [1] to compute gradients of observables with respect to the force field parameters used to estimate them.

This approach may be derived by direct differentiation of the ensemble average an observable X:

$$\langle X(\theta) \rangle = \frac{1}{Q(\theta)} \int X(\theta) \exp\left[-\beta \left(U(\vec{r}, V; \theta) + pV\right)\right] d\vec{r} dV$$

where

$$Q\left(\boldsymbol{\theta}\right) = \int \exp\left[-\beta\left(\boldsymbol{U}\left(\vec{r},\boldsymbol{V};\boldsymbol{\theta}\right) + p\boldsymbol{V}\right)\right]\mathrm{d}\vec{r}\mathrm{d}\boldsymbol{V}$$

is the isothermal-isobaric partion function, θ are the force field parameters being used to estimate the observable, U the systems potential energy, $\beta \equiv k_b T$, k_b the Boltzmann constant, T the temperature, p the pressure and V the volume.

The derivative of the ensemble average defined above with respect to a particular force field parameter of interest θ is given by:

$$\frac{\mathrm{d}\langle X\rangle}{\mathrm{d}\theta_i} = \left\langle \frac{\mathrm{d}X}{\mathrm{d}\theta_i} \right\rangle - \beta \left[\left\langle X \frac{\mathrm{d}U}{\mathrm{d}\theta_i} \right\rangle - \left\langle \frac{\mathrm{d}U}{\mathrm{d}\theta_i} \right\rangle \langle X \rangle \right\rangle$$

2.15.2 Computing $dU/d\theta_i$

While future integrations with differentiable simulation engines such as timemachine will allow $dU/d\theta_i$ to be computed directly from molecular simulation runs, currently most common simulation engines do not directly support computing this quantity.

Until such an integration is complete, the framework currently employs a central finite difference approach, whereby

$$\frac{\mathrm{d}U}{\mathrm{d}\theta_{i}} \approx \frac{U\left(\theta_{i}+h\right) - U\left(\theta_{i}-h\right)}{2h}$$

Although more expensive than computing either the forward or backwards derivative, the central difference method should give a more accurate estimate of the gradient at the minima, maxima and transition points. By default a value of $h = \theta_i \times 10^{-4}$ is used. This has been found to yield finite differences which do not suffer from precision issues, while being sufficiently small so as to yield an accurate estimate.

In practice the derivatives obtained by re-evaluating the energies of each configuration in a trajectory generated by a molecular simulation (either after a simulation or after loading one from disk) at each of the perturbed parameters.

While there is an expense associated with extra evaluations of the potential energy function for each configuration, this is mitigated by only computing those terms which depend upon (or may depend upon) θ_i . As an example, when computing derivatives with respect to a bond length the electrostatic and van der Waal contributions are not computed. This significantly speeds up the computation of these derivatives.

The final derivatives are stored in *ObservableArray* objects for convenience and for easy propagation of gradients through workflows. See the *observables documentation* for more information.

2.15.3 References

2.16 Calculation Layers

A *CalculationLayer* is an implementation of one calculation approach for estimating a set of physical properties, such as via molecular simulation or evaluating some QSAR like model.

The framework stacks multiple layers together when estimating a data set of properties.

Fig. 2: A schematic of the layer system. A set of properties to estimate are fed into the first layer. Those which can be calculated are returned back. Those that can't are passed to the next layer until no layer are left.

Each layer will in turn attempt to evaluate the properties being estimated using the specific approach the layer represents, such as by running a set of simulations. If the layer is unable to estimate a given property, for example if a layer does not yet support a given property, or if the layer has insufficient data to reprocesses, the property will be passed to the next layer for it to try and evaluate.

In practice, this allows the framework to attempt to estimate a data set using the most rapid calculation layer first, before moving to successively slower yet more robust layers, and thus enabling as efficient as possible property estimation.

2.16.1 Defining a Calculation Layer

A calculation layer is defined by two objects - a *CalculationLayer* object which implements the main layer logic, and a *CalculationLayerSchema* which defines those settings and options exposed required by the layer.

One *CalculationLayerSchema* will be provided to the for each type of property that the layer is being asked to estimate. The base *CalculationLayerSchema* currently only exposes options for optionally defining either the relative or absolute uncertainty that the layer should attempt to estimate the associated property type to within, however custom schemas can be defined per layer.

The structure of a *CalculationLayer* is relatively simple and permissive:

```
@calculation_layer()
class MyCalculationLayer(CalculationLayer):
    @classmethod
    def required_schema_type(cls):
        return CalculationLayerSchema
    @classmethod
    def _schedule_calculation(
            cls,
            calculation_backend,
            storage_backend,
            layer_directory,
            batch
    ):
    ....
```

The first thing to note is the *calculation_layer()* decorator which is being applied to the class. This registers the calculation layer with the frameworks plug-in system, allowing it to be used in future calculations.

The only other requirements is that the class implement a required_schema_type class method, which returns the type of *CalculationLayerSchema* that is associated with this layer, and a _schedule_calculation(). The _schedule_calculation() is responsible for performing the actual property calculations.

The form of the _schedule_calculation() function is very flexible:

```
@classmethod
def _schedule_calculation(
    cls,
    calculation_backend,
    storage_backend,
    layer_directory,
    batch
):
    futures = []
    for queued_property in batch.queued_properties:
        futures.append(
            calculation_backend.submit_task(
                cls.process_property, queued_property, cls.__name__
        )
```

)

return futures

It takes as arguments:

- a *CalculationBackend* which is used to asynchronously distribute any calculations across the available compute resources.
- a StorageBackend which may be used to store / cache any data generated by the calculations.
- the path to the directory within which all of the calculation working files should be stored.
- the *Batch* of properties which this layer should attempt to estimate. This object includes the properties to estimate, as well as the *CalculationLayerSchema* for each property type.

and must return a list of Future objects (which either must be or implement the same API as the asyncio Future object). The easiest way to generate the futures is to perform any calculations using the calculation_backend which will automatically return the results of any functions as such.

The future objects returned by _schedule_calculation() must return a *CalculationLayerResult* object, which includes

- the estimated property if the calculation was successful (or UNDEFINED otherwise).
- a list of any exceptions (of type EvaluatorException) which were raised during the calculation.
- a list of any data to be stored by the storage backend.

As a minimal example of a method which returns one such object:

```
@classmethod
def process_property(cls, physical_property, **_):
    """Return a result as if the property had been successfully estimated.
    """
    # TODO: Do some calculations
    # Set the property provenance
    physical_property.source = CalculationSource(fidelity=cls.__name__)
    # Return the results object.
    results = CalculationLayerResult()
    results.physical_property = physical_property
    return results
```

2.16.2 Default Schemas

Default schemas for each pair of a calculation layer and a type of physical property may be registered using the *register_calculation_schema()* function:

```
# Register the default schema to use for density measurements being estimated
# by the direct simulation calculation layer.
register_calculation_schema(
    property_class=Density,
    layer_class=SimulationLayer,
```

```
schema=Density.default_simulation_schema
```

where the schema object should either be an instance of a *CalculationLayerSchema*, or a function with no required arguments which returns a *CalculationLayerSchema*.

A list of the registered schemas is provided by the registered_calculation_schemas module attribute.

2.17 Workflow Layers

)

The WorkflowCalculationLayer and WorkflowCalculationSchema offer an abstract base implementation for any calculation layers (and their associated schemas) which will perform their calculations using the built-in workflow engine.

The WorkflowCalculationLayer takes as input from its calculation schema one WorkflowSchema object for each type of property to be estimated by this layer. These schemas must *at a minimum* provide both the schemas of the protocols in the workflow, and have the *final_value_source* attribute set to the value of the calculated observable. In addition, the layer fully supports schemas which provide gradient information (see the gradients_sources attribute), as well as storing any generated dataclasses (see the *outputs_to_store* attribute) to the available storage backend.

This layer implements three key methods which are available to be overridden by any subclass implementations:

- _get_workflow_metadata(): a method which returns the dictionary of *metadata* which will be made available to the workflow (see the *default metadata* section for details).
- _build_workflow_graph(): the method which will construct the *workflow graph* to execute using the input workflow schemas and the metadata generated by the layer.
- workflow_to_layer_result(): a method which will map any WorkflowResult objects generated by the workflow graph into the CalculationLayerResult objects which the layer requires.

The workflow layer will by default tag each property estimated using it (or one of its derivatives) with a *CalculationSource* with the *fidelity* attribute set to the name of the layer, and the *provenance* attribute set to the schema of the workflow used to generate the property.

2.17.1 Default Metadata

The metadata provided to the workflows generated by this layer is generated on a per property to estimate basis mainly using the *generate_default_metadata()* function. It includes:

Кеу	Туре	Description
thermodynamic_state	ThermodynamicState	The state at which the to perform any calculations .
substance	Substance	The substance to use in any calculations.
components	[Substance]	The components present in the main substance.
target_uncertainty	Quantity	The target uncertainty of any calculations defined by the calculation schema.
per_component_uncertainty	Quantity	The target_uncertainty divided by sqrt(substance.n_components + 1)
force_field_path	str	A file path to the force field parameters to use.
parameter_gradient_keys	[ParameterGradientKey]	The parameters to differentiate any observables with respect to (if any).

2.18 The Direct Simulation Layer

The *SimulationLayer* is a calculation layer which employs molecular simulation to estimate data sets of physical properties. It inherits the *WorkflowCalculationLayer* base layer, and primarily makes use of the built-in *workflow* engine to perform the required calculations.

The simulation layer is expected to *almost always* be able to estimate any properties requested of it (with exceptions being where a workflow schema has not yet been defined for a class of properties, or where an unexpected error occurs), and can be thought of as a safe 'fallback' layer when no other calculation approach are able to estimate particular properties.

It is expected that *workflow schemas* passed to the simulation layer should be able to estimate the gradients of the observable they aim to calculate, as well as specify a set of :doc:` storage/dataclasses <storage/dataclasses>` which contain the data generated by the molecular simulations.

2.18.1 Default Metadata

The simulation layer makes the same set of metadata available to its workflows as the parent workflow layer.

2.19 The MBAR Reweighting Layer

The *ReweightingLayer* is a calculation layer which employs the Multistate Bennett Acceptance Ratio (MBAR) method to calculate observables at states which have not been previously simulated, but for which simulations have been previously run at similar states and their data cached. It inherits the *WorkflowCalculationLayer* base layer, and primarily makes use of the built-in *workflow* engine to perform the required calculations.

Because MBAR is a technique which reprocesses exisiting simulation data rather than re-running new simulations, it is typically several fold faster than the *simulation layer* provided it has cached simulation data (made accessible via a *storage backend*) available. Any properties for which the required data (see *Calculation Schema*) is not available will be skipped.

2.19.1 Theory

The theory behind applying MBAR to reweighting observables from a simulated state to an unsimulated state is covered in detail in the publication Configuration-Sampling-Based Surrogate Models for Rapid Parameterization of Non-Bonded Interactions.

2.19.2 Calculation Schema

The reweighting layer will be provided with one *ReweightingSchema* per type of property that it is being requested to estimate. It builds off of the base *WorkflowCalculationSchema* schema providing an additional *storage_queries* attribute.

The *storage_queries* attribute will contain a dictionary of *SimulationDataQuery* which will be used by the layer to access the data required for each property from the storage backend. Each key in this dictionary will correspond to the key of a piece of metadata made available to the property workflows.

2.19.3 Default Metadata

The reweighting layer makes available the default metadata provided by the *parent workflow layer* in addition to any cached data retrieved via the schemas *storage_queries*.

When building the metadata for each property, a copy of the query will be made and any of the supported attributes (currently only *substance*) whose values are set as *PlaceholderValue* objects will have their values updated using values directly from the property. This query will then be passed to the storage backend to retrieve any matching data.

The matching data will be stored as a list of tuples of the form:

```
(object_path, data_directory, force_field_path)
```

where object_path is the file path to the stored dataclass, the data_directory is the file path to the ancillary data directory and force_field_path is the file path to the force field parameters which were used to generate the data originally.

This list of tuples will be made available as metadata under the key that was associated with the query.

2.20 Workflows

The framework offers a lightweight workflow engine for executing graphs of tasks using the available *calculation backends*. While lightweight, it offers a large amount of extensibility and flexibility, and is currently used by both the *simulation* and *reweighting* layers to perform their required calculations.

A workflow is a wrapper around a collection of tasks that should be executed in succession, and whose outputs should be made available as the input to others.

Fig. 3: A an example workflow which combines a protocol which will build a set of coordinates for a particular system, assign parameters to that system, and then perform an energy minimisation.

The workflow engine offers a number of advanced features such as the *automatic reduction of redundant tasks*, and *looping over parts of a workflow*

2.20.1 Building Workflows

At its core a workflow must define the tasks which need to be executed, and where the inputs to those tasks should be sourced from. Each task to be executed is represented by a *protocol object*, with each protocol requiring a specific set of user specified inputs:

```
# Define a protocol which will build some coordinates for a system.
build_coordinates = BuildCoordinatesPackmol("build_coordinates")
build_coordinates.max_molecules = 1000
build_coordinates.mass_density = 1.0 * unit.gram / unit.millilitre
build_coordinates.substance = Substance.from_components("0", "CO")
# Define a protocol which will assign force field parameters to the system.
assign_parameters = BuildSmirnoffSystem(f"assign_parameters")
assign_parameters.water_model = BuildSmirnoffSystem.WaterModel.TIP3P
assign_parameters.force_field_path = "openff-1.0.0.offxml"
```

Set the `coordinate_file_path` input of the `assign_parameters` protocol

```
# to the `coordinate_file_path` output of the `build_coordinates` protocol.
assign_parameters.coordinate_file_path = ProtocolPath(
    "coordinate_file_path", build_coordinates.id
)
```

The *ProtocolPath* object is used to reference the output of another protocol in the workflow, and will be replaced by the value of that output once that protocol has been executed by the workflow engine. It is constructed from two parts:

- the name of the output attribute to reference.
- the unique id of the protocol to take the output from.

To turn these tasks into a valid workflow which can be automatically executed, they must first be converted to a *workflow schema*:

```
# Create the schema object.
schema = WorkflowSchema()
# Add the individual protocol's schema representations to the workflow schema.
schema.protocol_schemas = [build_coordinates.schema, assign_parameters.schema]
# Create the executable workflow object from its schema.
workflow = Workflow.from_schema(schema, metadata=None)
```

A Workflow may either be synchronously executed in place yielding a WorkflowResult object directly:

```
workflow_result = workflow.execute()
```

or asynchronously using a calculation backend yielding a Future like object which will eventually return a *WorkflowResult*:

```
with DaskLocalCluster() as calculation_backend:
    result_future = workflow.execute(calculation_backend=calculation_backend)
```

In addition, a workflow may be add to, and executed as part as a larger workflow graphs.

2.20.2 Workflow Schemas

A *WorkflowSchema* is a blueprint from which all *Workflow* objects are constructed. It will predominantly define the tasks which compose the workflow, but may optionally define:

- *final_value_source*: A reference to the protocol output which corresponds to the value of the main observable calculated by the workflow.
- gradients_sources: A list of references to the protocol outputs which correspond to the gradients of the main observable with respect to a set of force field parameters.
- outputs_to_store: A list of data classes whose values will be populated from protocol outputs.
- *protocol_replicators*: A set of *replicators* which are used to flag parts of a workflow which should be replicated.

Each of these attributes will control whether the *value*, *gradients* and *data_to_store* attributes of the *WorkflowResult* results object will be populated respectively when executing a workflow.

Metadata

Because a schema is purely a blueprint for a general workflow, it need not define the exact values of all of the inputs of its constituent tasks. Consider the above example workflow for constructing a set of coordinates and assigning force field parameters to them. Ideally this one schema could be reused for multiple substances. This is made possible through a workflows *metadata*.

Each protocol within a workflow may access a dictionary of values unique to that workflow (termed here *metadata*) which is defined when the *Workflow* object is created from its schema.

This metadata may be accessed by protocols via a fictitious "global" protocol whose outputs map to the metadata dictionary:

```
build_coordinates = BuildCoordinatesPackmol("build_coordinates")
build_coordinates.substance = ProtocolPath("substance", "global")
# ...
substances = [
    Substance.from_components("CO"),
    Substance.from_components("CCO"),
    Substance.from_components("CCCO"),
]
for substance in substances:
    # Define the metadata to make available to the workflow protocols.
    metadata = {"substance": substance}
    # Create the executable workflow object from its schema.
    workflow = Workflow.from_schema(schema, metadata=metadata)
    # Execute the workflow ...
```

the created workflow will contain the build_coordinates protocol but with its substance input set to the value from the metadata dictionary.

2.21 Replicators

A *ProtocolReplicator* is the workflow equivalent of a for loop. It is statically evaluated when a *Workflow* is created from its schema. This is useful when parts of a workflow should be run multiple times but using different values for certain protocol inputs.

Note: The syntax of replicators is still rather rough around the edges, and will be refined in future versions of the framework.

Each *ProtocolReplicator* requires both a unique id and the set of *template values* which the replicator will 'loop' over to be defined. These values must either be a list of constant values or a reference to a list of values provided as *metadata*.

The 'loop variable' is referenced by protocols in the workflow using the *ReplicatorValue* placeholder input, where the value is linked to the replicator through the replicators unique id.

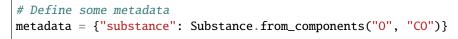
As an example, consider the case where a set of coordinates should be built for each component in a substance:

```
# Create the replicator object, and assign it a unique id.
replicator = ProtocolReplicator(replicator_id="component_replicator")
# Instruct the replicator to loop over all of the components of the substance
# made available by the global metadata
replicator.template_values = ProtocolPath("substance.components", "global")
# Define a protocol which will build some coordinates for a system.
build_coords = BuildCoordinatesPackmol("build_coords_" + replicator.placeholder_id})
# Instruct the protocol to use the value specified by the replicator.
build_coords.substance = ReplicatorValue(replicator.id)
# Build the schema containing the protocol and the replicator
schema = WorkflowSchema()
schema.protocol_schemas = [build_coords.schema]
schema.protocol_replicators = [replicator]
```

The requirement for a protocol to be replicated by a replicator is that its id *must* contain the replicators *placeholder_id* - this is a simple string which the workflow engine looks for when applying the replicator. The contents of this schema can be easily inspected by printing its JSON representation:

```
{
    "@type": "openff.evaluator.workflow.schemas.WorkflowSchema",
    "protocol_replicators": [
        {
            "@type": "openff.evaluator.workflow.schemas.ProtocolReplicator",
            "id": "component_replicator",
            "template_values": {
                "@type": "openff.evaluator.workflow.utils.ProtocolPath",
                "full_path": "global.substance.components"
            }
        }
   ],
    "protocol_schemas": [
        {
            "@type": "openff.evaluator.workflow.schemas.ProtocolSchema",
            "id": "build_coords_$(component_replicator)",
            "inputs": {
                ".substance": {
                    "@type": "openff.evaluator.workflow.utils.ReplicatorValue",
                    "replicator_id": "component_replicator"
                }
            },
            "type": "BuildCoordinatesPackmol"
        }
   ]
}
```

It can be clearly seen that the schema only contains a single protocol entry, with the placeholder id present in its unique id. Once a workflow is created from this schema however:



```
# Build the workflow from the schema.
workflow = Workflow.from_schema(schema, metadata)
# Output the contents of the workflow as JSON.
print(workflow.schema.json())
```

it can be seen that the replicator has been correctly been applied and the workflow now contains one protocol for each component in the substance passed as metadata:

```
{
    "@type": "openff.evaluator.workflow.schemas.WorkflowSchema",
    "protocol_schemas": [
        {
            "@type": "openff.evaluator.workflow.schemas.ProtocolSchema",
            "id": "build_coords_0",
            "inputs": {
                ".substance": {
                    "@type": "openff.evaluator.substances.components.Component",
                    "smiles": "O"
                }
            },
            "type": "BuildCoordinatesPackmol"
        },
        {
            "@type": "openff.evaluator.workflow.schemas.ProtocolSchema",
            "id": "build_coords_1".
            "inputs": {
                ".substance": {
                    "@type": "openff.evaluator.substances.components.Component",
                    "smiles": "CO"
                }
            },
            "type": "BuildCoordinatesPackmol"
        }
    ]
}
```

In both cases the replicators *placeholder_id* has been replaced with the index of the value it was replicated for, and the substance input has been correctly set to the actual array value.

2.21.1 Nested Replicators

Replicators can be applied to other replicators to achieve a result similar to a set of nested for loops. For example the below loop:

```
components = [Component("0"), Component("CO")]
n_mols = [[1000], [500]]
for i, component in enumerate(components):
    for component_n_mols in n_mols[i]:
        ...
```

can readily be reproduced using replicators:

```
# Define a replicator which will loop over all components in the substance.
component_replicator = ProtocolReplicator(replicator_id="components")
component_replicator.template_values = ProtocolPath("components", "global")
# Define a replicator to loop over the number of each component to add.
n_mols_replicator_id = f"n_mols_{component_replicator.placeholder_id}"
n_mols_replicator = ProtocolReplicator(replicator_id=n_mols_replicator_id)
n_mols_replicator.template_values = ProtocolPath(
    f"n_mols[{component_replicator.placeholder_id}]", "global"
)
# Define the suffix which must be applied to protocols to be replicated
id_suffix = f"{component_replicator.placeholder_id}_{n_mols_replicator.placeholder_id}"
# Define a protocol which will build some coordinates for a system.
build_coordinates = BuildCoordinatesPackmol(f"build_coordinates_{id_suffix}")
build_coordinates.substance = ReplicatorValue(component_replicator.id)
build_coordinates.max_molecules = ReplicatorValue(n_mols_replicator.id)
# Build the schema containing the protocol and the replicator
schema = WorkflowSchema()
schema.protocol_schemas = [build_coordinates.schema]
schema.protocol_replicators = [component_replicator, n_mols_replicator]
# Define some metadata
metadata = {
    "components": [Component("0"), Component("CO")],
    "n_mols": [[1000], [500]]
}
# Build the workflow from the created schema.
workflow = Workflow.from_schema(schema, metadata)
# Print the JSON representation of the workflow.
print(workflow.schema.json(format=True))
```

Here the component_replicator placeholder id has been appended to the n_mols_replicator id to inform the workflow engine that the later is a child of the former. The component_replicator placeholder id is then used as an index into the n_mols array. This results in the following schema as desired:

```
},
            "type": "BuildCoordinatesPackmol"
        },
        {
            "@type": "openff.evaluator.workflow.schemas.ProtocolSchema",
            "id": "build_coordinates_1_0",
            "inputs": {
                ".max_molecules": 500,
                ".substance": {
                     "Otype": "openff.evaluator.substances.components.Component",
                     "smiles": "CO"
                }
            },
            "type": "BuildCoordinatesPackmol"
        }
    ]
}
```

2.22 Workflow Graphs

A *WorkflowGraph* is a collection of *Workflow* objects which should be executed together. The primary advantage of executing workflows via the graph object is that the graph will automatically take advantage of the *protocols* built in redundancy / merging support to collapse duplicate tasks across multiple workflows.

As an example, consider the case of executing workflows to estimate the density and the dielectric constant at the same state point, for the same substance, and using the same force field parameters:

```
density_schema = Density.default_simulation_schema()
dielectric_schema = DielectricConstant.default_simulation_schema()
density_workflow = Workflow.from_schema(density_schema, metadata)
dielectric_workflow = Workflow.from_schema(dielectric_schema, metadata)
print(len(density_workflow.protocols), len(dielectric_workflow.protocols))
workflow_graph = WorkflowGraph()
workflow_graph.add_workflows(density_workflow, dielectric_workflow)
print(len(workflow_graph.protocols))
```

The final workflow graph has roughly half the total number of density and dielectric protocols to be executed. This is expected as both the density and dielectric workflows are almost identical, except for the final analysis steps.

Graphs can be executed either in place without using a calculation backend in the same way that workflows can.

2.23 Protocols

The *Protocol* class represents a single task to be executed, whether that be as a standalone task or as a task which is part of some larger workflow. The task encoded by a protocol may be as simple as adding two numbers together or even as complex as performing entire free energy simulations:

```
from openff.evaluator.protocols.miscellaneous import AddValues
# Create the protocol and assign it some unique name.
add_numbers = AddValues(protocol_id="add_values")
# Set the numbers to add together
add_numbers.values = [1, 2, 3, 4]
# Execute the protocol
add_numbers.execute()
# Retrieve the output
result = add_numbers.result
```

2.23.1 Inputs and Outputs

Each protocol exposes a set of the required inputs as well as the produced outputs. These inputs may either be set as a constant directly, or if used as part of a *workflow*, can take their value from one of the outputs of another protocol.

Fig. 4: A selection of the inputs and outputs of the OpenMMSimulation protocol.

A surprisingly rich spectrum of workflows can be constructed by chaining together many relatively simple protocols.

The inputs and outputs of a protocol are defined using the custom *InputAttribute* and *OutputAttribute* descriptors:

```
class AddValues(Protocol):
    # Define the inputs that the protocol requires
   values = InputAttribute(
        docstring="The values to add together.",
        type_hint=list, default_value=UNDEFINED
   )
   # Define the outputs that the protocol will produce
   # once it is executed.
   result = OutputAttribute(
       docstring="The sum of the values.",
        type_hint=typing.Union[int, float, unit.Measurement, unit.Quantity],
   )
   def _execute(self, directory, available_resources):
        . . .
   def validate(self, attribute_type=None):
        . . .
```

Here we have defined a values input to the protocol and a result output. Both descriptors require a docstring and a type_hint to be provided.

The type_hint will be used by the workflow engine to ensure that a protocol which takes its input as the output of another protocol is receiving values of the correct type. Currently the type_hint can be any type of python class, or a Union of multiple types should the protocol allow for that.

In addition, the input attribute must specify a default_value for the attribute. This can either be a constant value, or a value set by some function such as a lambda statement:

```
some_input = InputAttribute(
    docstring="Takes it's default value from a function.",
    type_hint=int,
    default_value=lambda: return 1 + 1
)
```

In the above example we set the default value of values to UNDEFINED in order to specify that this input must be set by the user. The custom UNDEFINED class is used in place of None as None may be a valid input value for some attributes.

2.23.2 Task Execution

In addition to defining its inputs and outputs, a protocol must also implement an _execute() function which handles the main logic of the task:

```
def _execute(self, directory, available_resources):
    self.result = self.values[0]
    for value in self.values[1:]:
        self.result += value
```

The function is passed the directory in which it should run and create any working files, as well as a *ComputeResources* object which describes which compute resources are available to run on. This function *must* set all of the output attributes of the protocol before returning.

The private _execute() function which must be implemented should not be confused with the public execute() function. The public execute() function implements some common protocol logic (such as validating the inputs and creating the directory to run in if needed) before calling the private _execute() function.

2.23.3 Protocol Validation

The protocols inputs will automatically be validated before _execute() is called - this validation includes making sure that all of the non-optional inputs have been set, as well as ensuring they have been set to a value of the correct type. Protocols may implement additional validation logic by implementing a *execute()* function:

```
def validate(self, attribute_type=None):
    super(AddValues, self).validate(attribute_type)
    if len(self.values) < 1:
        raise ValueError("There were no values to add together")</pre>
```

2.23.4 Schemas

Every protocol has a *ProtocolSchema* representation which uniquely describes the protocol, and from which the protocol can be exactly recreated. The schema stores not only the type of protocol which it represents, but also the values of each of the inputs. Protocol schemas are fully JSON serializable. The following is an example schema for the above add_numbers protocol:

```
{
  "@type": "openff.evaluator.workflow.schemas.ProtocolSchema",
  "id": "add_values",
  "inputs": {
    ".allow_merging": true,
    ".values": [1, 2, 3, 4]
  },
  "type": "AddValues"
}
```

A protocols schema can be accessed via it's *schema* attribute. A protocol can be directly created from its schema representation by calling the schema's *to_protocol()* function.

2.23.5 Merging Protocols

When executing multiple workflows together (e.g. executing a workflow to estimate a substances density and potential energy) there is a large likelihood that some of tasks in those two workflows will be identical. Examples may include two workflows requiring protocols which build a set of coordinates, or assigning the same set of parameters to those coordinates.

Protocols have built-in support for comparing whether they are performing the same task / calculation as another protocol through the *can_merge()* and *merge()* functions:

- The *can_merge()* function checks to see whether two protocols are performing an identical task and hence whether they should be merged or not.
- The merge() function handles the actual merging of two protocols which can be merged.

The default *can_merge()* function takes advantage of the merge_behvaiour attribute of the different input descriptors. The merge_behvaiour attribute describes how each input should be considered when checking to see if two protocols can be merged:

```
max_molecules = InputAttribute(
    docstring="The maximum number of molecules to be added to the system.",
    type_hint=int,
    default_value=1000,
    merge_behavior=MergeBehaviour.ExactlyEqual
)
```

The most common behavior is to require that the inputs must be ExactlyEqual in order for two protocols two be considered to be identical. However, for some inputs such as the timestep of a simulation or the number of steps to simulate for, the exact values of the inputs don't necessarily need to be equal but rather, we may just wish to take the larger / smaller of the two inputs:

```
timestep = InputAttribute(
    docstring="The timestep to evolve the system by at each step.",
    type_hint=unit.Quantity,
    merge_behavior=InequalityMergeBehaviour.SmallestValue,
```

```
default_value=2.0 * unit.femtosecond,
)
total_number_of_iterations = InputAttribute(
    docstring="The number of times to propogate the system forward by.",
    type_hint=int,
    merge_behavior=InequalityMergeBehaviour.LargestValue,
    default_value=1,
)
```

This can be accomplished using the InequalityMergeBehaviour enum.

The default *merge()* function also relies upon the *merge_behaviour* attributes to determine which values of the inputs should be retained when merging two protocols.

2.24 Protocol Groups

The *ProtocolGroup* class represents a collection of *protocols* which have been grouped together. All protocols within a group will be executed together on a single compute resources, i.e. there is currently no support for executing protocols within a group in parallel.

Protocol groups have a specialised *ProtocolGroupSchema* which is essentially a collection of *ProtocolSchema* objects.

2.24.1 Conditional Protocol Groups

A *ConditionalGroup* is a special class of *ProtocolGroup* which will execute all of the grouped protocols again and again until a set of conditions has been met or until a maximum number of iterations (see *max_iterations*) has been performed. They can be thought of as being a protocol representation of a while statement.

Each condition to be met is represented by a *Condition* object:

```
condition = ConditionalGroup.Condition()
# Set the left and right hand values.
condition.left_hand_value = ...
condition.right_hand_value = ...
# Choose the type of condition
condition.type = ConditionalGroup.Condition.Type.LessThan
```

The left and right hand values can either be constants, or come from the output of another protocol (including grouped protocols) using a *ProtocolPath*. Currently a condition can either check that a value is less than or greater than another value.

Conditional groups expose a *current_iteration* attribute which tracks how many times the grouped protocols have been executed. This can be used as input by any of the grouped protocols and is useful, for example, to run a simulation for longer and longer until the groups condition has been met:

```
conditional_group = ConditionalGroup("conditional_group")
```

```
# Set up protocols to run a simulation and then to extract the
```

```
# value of the density and its uncertainty.
simulation = OpenMMSimulation("simulation")
simulation.input_coordinate_file = "coords.pdb"
simulation.parameterized_system = ...
extract_density = AverageObservable("extract_density")
extract_density.observable = simulation.observables["Density"]
# Set the total number of iterations the simulation should perform to be equal
# to the current iteration of the group. I.e the simulation should perform a
# new iteration at each group iteration.
simulation.total_number_of_iterations = ProtocolPath(
    "current_iteration", conditional_group.id
)
# Add the protocols to the group.
conditional_group.add_protocols(production_simulation, analysis_protocol)
# Set up a condition which will check if the uncertainty is less than
# some threshold.
condition = ConditionalGroup.Condition()
condition.condition_type = groups.ConditionalGroup.Condition.Type.LessThan
condition.right_hand_value = 0.5 * unit.gram / unit.millilitre
condition.left_hand_value = ProtocolPath(
    "value.error", conditional_group.id, analysis_protocol.id
)
# Add the condition.
conditional_group.add_condition(condition)
```

It is this idea which is used to continue running a molecular simulations until an observable of interest (such as the density) has been calculated to within a specified uncertainty.

2.25 Observables

A key feature of this framework is its ability to compute the gradients of physical properties with respect to the force field parameters used to estimate them. This requires the framework be able to, internally, be able to not only track the gradients of all quantities which combine to yield the final observable of interest, but to also be able to propagate the gradients of those composite quantities through to the final value.

The framework offers three such objects to this end (*Observable*, *ObservableArray* and *ObservableFrame* objects) which will be covered in this document.

Note: In future versions of the framework the objects described here will likely be at least in part deprecated in favour of using full automatic differentiation libraries such as jax. Supporting these libraries will take a large re-write of the framework however, as well as full support between differentiable simulation engines like timemachine and the OpenFF toolkit. As such, these objects are implemented as stepping stones which can be gently phased out while working towards that larger, more modern goal.

2.25.1 Observable Objects

The base object used to track observables is the *Observable* object. It stores the average value, the standard error in the value and the gradient of the value with respect to force field parameters of interest.

Currently the value and error are internally stored in a composite Measurement object, which themselves wrap around the uncertainties package. This allows uncertainties to be automatically propagated through operations without the need for user intervention.

Note: Although uncertainties are automatically propagated, it is still up to property estimation workflow authors to ensure that such propagation (assuming a Gaussian error model) is appropriate. An alternative, which is employed throughout the framework is to make use of the bootstrapping technique.

Gradients are stored in a list as *ParameterGradient* gradient objects, which store both the floating value of the gradient alongside an identifying *ParameterGradientKey*.

Supported Operations

- + and -: *Observable* objects can be summed with and subtracted from other *Observable* objects, Quantity objects, floats or integers. When two *Observable* objects are summed / subtracted, their gradients are combined by summing / subtracting also. When an *Observable* is summed / subtracted with a Quantity, float or int object it is assumed that these objects do not depend on any force field parameters.
- *: *Observable* objects may be multiplied by other *Observable* objects, Quantity objects, and float or int objects. When two *Observable* objects are multiplied their gradients are propagated using the product rule. When an *Observable* is multiplied by a Quantity, float or int object it is assumed that these objects do not depend on any force field parameters.
- *I*: *Observable* objects may be divided by other *Observable* objects, Quantity objects, and float or int objects. Gradients are propagated through the division using the quotient rule. When an *Observable* is divided by a Quantity, float or int object (or when these objects are divided by an *Observable* object) it is assumed that these objects do not depend on any force field parameters.

In all cases two *Observable* objects can only be operated on provided the contain gradient information with respect to the same set of force field parameters.

2.25.2 Observable Arrays

An extension of the *Observable* object is the *ObservableArray* object. Unlike an *Observable*, an *ObservableArray* object does not contain error information, but rather the value it stores and the gradients of that value should be a numpy array with shape=(n_data_points, n_dimensions). It is designed to store information such as the potential energy evaluated at each configuration sampled during a simulation, as well as the gradient of the potential, which can then be ensemble averaged using a fluctuation formula to propagate the gradients through to the average.

Like with *Observable* objects, gradients are stored in a list as *ParameterGradient* gradient objects. The length of the gradients is required to match the length of the value array.

ObservableArray objects may be concatenated together using their join() method or sub-sampled using their subset() method.

Supported Operations

The *ObservableArray* object supports the same operations as the *Observable* object, whereby all operations are applied elementwise to the stored arrays.

2.25.3 Observable Frames

An *ObservableFrame* is a wrapper around a collection of *ObservableArray* which contain the types of observable specified by the *ObservableType* enum. It behaves as a dictionary which can take either an *ObservableType* or a string value of an *ObservableType* as an index.

Like an *ObservableArray*, observable frames may be concatenated together using their *join()* method or sub-sampled using their *subset()* method.

Supported Operations

No operations are supported between observable frames.

submit_task

2.26 Calculation Backends

A *CalculationBackend* is an object used to distribute calculation tasks across available compute resources. This is possible through specific backends which integrate with libraries such as multiprocessing, dask, parsl and cerlery.

Each backend is responsible for creating *compute workers*. A compute worker is an entity which has a set amount of dedicated compute resources available to it and which can execute python functions using those resources. Calculation backends may spawn multiple workers such that many tasks and calculations can be performed simultaneously.

A compute worker can be as simple as a new multiprocessing **Process** or something more complex like a dask worker. The resources available to a worker are described by the *ComputeResources* object.

CalculationBackend classes have a relatively simple structure:

```
class MyCalculationBackend(CalculationBackend):
    def __init__(self, number_of_workers, resources_per_worker):
        ...
    def start(self):
        ...
    def stop(self):
        ...
    def submit_task(self, function, *args, **kwargs):
        ...
```

By default they implement a constructor which takes as input the number of workers that the backend should initially spawn as well as the compute resources which are available to each. They must further implement:

• a *start()* method which spawns the initial set of compute workers.

- a *stop()* method which should kill all workers spawned by the backend as well as cleanup any temporary worker files.
- a *submit_task()* method which takes a function to be execute by a worker, and a set of args and kwargs to pass to that function.

The *submit_task()* must run asynchronously and return an asyncio Future object (or an object which implements the same API) when called, which can then be queried for when the task has completed.

All calculation backends are implemented as context managers such that they can be used as:

```
with MyCalculationBackend(number_of_workers=..., resources_per_worker...) as backend:
    backend.submit_task(...)
```

where the *start()* and *stop()* methods will be called automatically.

2.27 Dask Backends

The framework implements a number of calculation backends which integrate with the **dask** distributed and job-queue libraries.

2.27.1 Dask Local Cluster

The DaskLocalCluster backend wraps around the dask LocalCluster class to distribute tasks on a single machine:

Its main purpose is for use when debugging calculations locally, or when running calculations on machines with large numbers of CPUs or GPUs.

2.27.2 Dask HPC Cluster

The *DaskLSFBackend* and *DaskPBSBackend* backends wrap around the dask LSFCluster and PBSCluster classes respectively, and both inherit the *BaseDaskJobQueueBackend* class which implements the core of their functionality. They predominantly run in an adaptive mode, whereby the backend will automatically scale up or down the number of workers based on the current number of tasks that the backend is trying to execute.

These backends integrate with the queueing systems which most HPC cluster use to manage task execution. They work by submitting jobs into the queueing system which themselves spawn dask workers, which in turn then execute tasks on the available compute nodes:

```
# Create the object which describes the compute resources each worker should request from
# the queueing system.
```

(continues on next page)

(continued from previous page)

```
worker_resources = QueueWorkerResources(
   number_of_threads=1,
   number_of_gpus=1,
   preferred_gpu_toolkit=QueueWorkerResources.GPUToolkit.CUDA,
   per_thread_memory_limit=worker_memory,
   wallclock_time_limit="05:59",
)
# Create the backend object.
setup_script_commands = [
    f"conda activate evaluator",
    f"module load cuda/10.1",
]
calculation_backend = DaskLSFBackend(
   minimum_number_of_workers=1,
   maximum_number_of_workers=max_number_of_workers,
   resources_per_worker=queue_resources,
   queue_name="gpuqueue",
    setup_script_commands=setup_script_commands,
)
# Perform some tasks.
with calculation_backend:
   calculation_backend.submit_task(logging.info, "Hello World")
    . . .
```

The setup_script_commands argument takes a list of commands which should be run by the queue job submission script before spawning the actual worker. This enables setting up custom environments, and setting any required environmental variables.

Configuration

To ensure optimal behaviour we recommend changing / uncommenting the following settings in the dask distributed configuration file (this can be found at ~/.config/dask/distributed.yaml):

```
distributed:
```

```
worker:
    daemon: False
comm:
    timeouts:
        connect: 10s
        tcp: 30s
deploy:
        lost-worker-timeout: 15s
```

See the dask documentation for more information about changing dask settings.

2.28 Storage Backends

A *StorageBackend* is an object used to store data generated as part of property calculations, and to retrieve that data for use in future calculations.

In general, most data stored in a storage backend is stored in two parts:

- A JSON serialized representation of this class (or a subclass), which contains lightweight information such as the state and composition of a system.
- 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, such as simulation trajectories, whose files are referenced by their file name by the data object.

The ancillary directory-like structure is not required if the data may be suitably stored in the data object itself.

2.28.1 Data Storage / Retrieval

Each piece of data which is stored in a backend must inherit from the *BaseStoredData* class, will be assigned a unique key. This unique key is both useful for tracking provenance if this data is re-used in future calculations, and also can be used to retrieve the piece of data from the storage system.

In addition to retrieval using the data keys, each backend offers the ability to perform a 'query' to retrieve data which matches a set of given criteria. Data queries are implemented via *BaseDataQuery* objects, which expose different options for querying for specific types of data (such a simulation data, trained models, etc.).

A query may be used for example to match all simulation data that was generated for a given *Substance* in a particular phase:

```
# Look for all simulation data generated for liquid water
substance_query = SimulationDataQuery()
substance_query.substance = Substance.from_components("0")
substance_query.property_phase = PropertyPhase.Liquid
found_data = backend.guery(substance_guery)
```

The returned found_data will be a dictionary with keys of tuples and values as lists of tuples. Each key will be a tuple of the values which were matched, for example the matched thermodynamic state, or the matched substance. For each value tuple in the tuple list, the first item in the tuple is the unique key of the found data object, the second item is the data object itself, and the final object is the file path to the ancillary data directory (or None if none is present).

See the Data Classes and Queries page for more information about the available data classes, queries and their details.

2.28.2 Implementation

A StorageBackend must at minimum implement a structure of:

(continues on next page)

(continued from previous page)

```
def _object_exists(self, storage_key):
```

where

. . .

- _store_object() must store a *BaseStoredData* object as well as optionally its ancillary data directory, and return a unique key assigned to that object.
- _retrieve_object() must return the *BaseStoredData* object which has been assigned a given key if the object exists in the system, as well as the file path to ancillary data directory if it exists.
- _object_exists() should return whether any object still exists in the storage system with a given key.

All of these methods will be called under a reentrant thread lock and may be considered as thread safe.

2.29 Data Classes and Queries

All data which is to be stored within a *StorageBackend* must inherit from the *BaseStoredData* class. More broadly there are typically two types of data which are expected to be stored:

- *HashableStoredData* data which is readily hashable and can be quickly queried for in a storage backend. The prime examples of such data are *ForceFieldData*, whose hash can be easily computed from the file representation of a force field.
- ReplaceableData data which should be replaced in a storage backend when new data of the same type, but which has a higher information content, is stored in the backend. An example of this is when storing a piece of StoredSimulationData in the backend which was generated for a particular Substance and at the same ThermodynamicState as an existing piece of data, but which stores many more uncorrelated configurations.

Every data class **must** be paired with a corresponding data query class which inherits from the *BaseDataQuery* class. In addition, each data object must implement a *to_storage_query()* function which returns the data query which would uniquely match that data object. The *to_storage_query()* is used heavily by storage backends when checking if a piece of data already exists within the backend.

2.29.1 Force Field Data

The *ForceFieldData* class is used to *ForceFieldSource* objects within the storage backend. It is a hashable storage object which allows for rapidly checking whether any calculations have been previously been performed for a particular force field source.

It has a corresponding *ForceFieldQuery* class which can be used to query for particular force field sources within a storage backend.

2.29.2 Cached Simulation Data

Classes derived from the *BaseSimulationData* class are used to store the data generated by molecular simulation. The data object primarily records the *Substance*, *PropertyPhase* and *ThermodynamicState* that the simulation was run at, as well as provenance about the calculation and the force field parameters used (as the key of the force field in the storage system).

It has a corresponding *BaseSimulationDataQuery* class which can be used to query for simulation data which matches a set of particular criteria within a storage backend, which in part includes querying for data collected:

- at a given thermodynamic_state (i.e temperature and pressure).
- for a given property_phase (e.g. gas, liquid, liquid+gas coexisting, ...).
- using a given set of force field parameters identified by their unique force_field_id assigned by the storage system

Additionally included is not only the ability to find data generated for a particular substance (e.g. only data for methanol), but also the ability to return data for each component of a given substance by setting the *substance_query* attribute to a *SubstanceQuery* which has the *components_only* attribute set to true:

```
# Load an existing storage backend
storage_backend = LocalFileStorage()
# Define a system of 50% water and 50% methanol.
full_substance = Substance.from_components("0", "CO")
# Look for all simulation data generated for the full substance
data_query = SimulationDataQuery()
data_query.substance = full_substance
data_query.property_phase = PropertyPhase.Liquid
full_substance_data = storage_backend.query(data_query)
# Now look for all of the pure data which has been stored for both pure
# water and pure methanol.
pure_substance_query = SubstanceQuery()
pure_substance_query.components_only = True
data_query.substance_query = pure_substance_query
component_data = storage_backend.query(data_query)
```

This is particularly useful for when retrieving data for use in the calculation of excess properties (such as the enthalpy of mixing), where such calculations require information about both the full mixture as well as the pure components.

Single Simulation Data

The *StoredSimulationData* class is used to store data generated by a *single* molecular simulation and can be queried for using its accompanying *SimulationDataQuery* query class. In addition to the data stored by the parent *BaseSimulationData* class, this class further stores:

- the number of molecules which were simulated.
- the topology of the simulated system (stored as ancillary data).
- and trajectory of configurations (stored as ancillary data) and observables generated by the simulation.
- the statistic inefficiency of the data.

Data of this kind is considered replaceable, whereby data which has the lowest statistical efficiency is preferred. The philosophy here is that we should store the maximum amount of samples (i.e the maximum number of uncorrelated samples for the property which has the shortest correlation time) which will be useful for future calculations, such that future calculations can simply discard the data which cannot be used (i.e. is likely correlated).

Free Energy Data

The *StoredFreeEnergyData* class is used to store data generated by a free energy calculation which computes the free energy difference between an end and start state. It can be queried for using its accompanying *FreeEnergyDataQuery* query class.

In addition to the data stored by the parent *BaseSimulationData* class, this class further stores:

- the free energy difference between the end and starting state.
- the topology of the system (stored as ancillary data).
- and trajectory of configurations generated in the starting and end states (stored as ancillary data).

Although data of this kind inherits from the *ReplaceableData* base class, all data deposited in a storage backend will be retained. At this time no situation can be envisaged that the same free energy data from exactly the same calculation will be stored, with the exception of operator errors.

2.30 Local File Storage

The *LocalFileStorage* backend stores and retrieves all data objects to / from the local file system. The root directory in which all data is to be stored is defined when the object is created:

```
storage_backend = LocalFileStorage(root_directory="stored_data")
```

All data objects will be stored within this directory as JSON files, with file names of the storage key assigned to that object. If the data object has an associated ancillary data directory, this will be **moved** (not copied) into the root directory and renamed to the storage key when that object is stored into the system.

An example directory created by a local storage backend will look something similar to:

```
    root_directory
    1fe615c5cb48429ab77fd71125dec297
    trajectory.dcd
    statistics.csv
```

- 3e15d19e0e614d0491a1a0bc9a51534e

(continues on next page)

(continued from previous page)

```
trajectory.dcd
statistics.csv
1fe615c5cb48429ab77fd71125dec297.json
3e15d19e0e614d0491a1a0bc9a51534e.json
0f71f2b4a22042d89d6f0882406869b6.json
```

where here the backend contains two data objects with ancillary data directories, and one without.

When retrieving data which has an ancillary data directory from the backend, the returned directory path will be the full path to the directory in the root storage directory.

2.31 Building the Docs

Although documentation for the OpenFF Evaluator is readily available online, it is sometimes useful to build a local version such as when

- developing new pages which you wish to preview without having to wait for ReadTheDocs to finish building.
- debugging errors which occur when building on ReadTheDocs.

In these cases, the docs can be built locally by doing the following:

```
git clone https://github.com/openforcefield/openff-evaluator.git
cd openff-evaluator/docs
conda env create --name openff-evaluator-docs --file environment.yaml
conda activate openff-evaluator-docs
rm -rf api && make clean && make html
```

The above will yield a new directory named *_build* which will contain the built html files which can be viewed in your local browser.

2.32 API

Documentation for each of the classes contained within the *openff.evaluator* framework.

2.32.1 Client Side API

EvaluatorClient	The object responsible for connecting to, and submit-
	ting physical property estimation requests to an Evalua-
	torServer.
BatchMode	The different modes in which a server can batch together
	properties to estimate.
ConnectionOptions	The options to use when connecting to an Evalua-
	torServer
Request	An estimation request which has been sent to a Evalua-
	torServer instance.
RequestOptions	The options to use when requesting a set of physical
	properties be estimated by the server.
	continues on next page

Table 2 – continued from previous page	
RequestResult	The current results of an estimation request - these re-
	sults may be partial if the server hasn't yet completed
	the request.

EvaluatorClient

class openff.evaluator.client.EvaluatorClient(connection options=None)

The object responsible for connecting to, and submitting physical property estimation requests to an *Evalua-torServer*.

Examples

These examples assume that an *EvaluatorServer* has been set up and is running (either synchronously or asynchronously). This server can be connect to be creating an *EvaluatorClient*:

```
>>> from openff.evaluator.client import EvaluatorClient
>>> property_estimator = EvaluatorClient()
```

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

```
>>> from openff.evaluator.client import ConnectionOptions
>>>
connection_options = ConnectionOptions(server_address='server_address',
>>> server_port=8000)
>>> property_estimator = EvaluatorClient(connection_options)
```

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

```
>>> # Load in the data set of properties which will be used for comparisons
>>> from openff.evaluator.datasets.thermoml 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 openff.evaluator import unit
>>> from openff.evaluator.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 openff.evaluator.forcefield import SmirnoffForceFieldSource
>>> force_field_source = SmirnoffForceFieldSource.from_path('smirnoff99Frosst-1.1.0.
\rightarrow offxml')
>>>
>>> # Submit the estimation request to a running server.
>>> 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 *RequestOptions* object to the estimate commands.

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

```
>>> from openff.evaluator.layers.reweighting import ReweightingLayer
>>> from openff.evaluator.layers.simulation import SimulationLayer
>>> options = RequestOptions(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 by providing a calculation schema to the options object.

```
>>> from openff.evaluator.properties import DielectricConstant
>>>
>>> # Generate a schema to use when estimating densities directly
>>> # from simulations.
>>> density_simulation_schema = Density.default_simulation_schema()
>>> # Generate a schema to use when estimating dielectric constants
>>> # from cached simulation data.
>>> dielectric_reweighting_schema = DielectricConstant.default_reweighting_schema()
>>>
>>> options.workflow_options = {
        'Density': {'SimulationLayer': density_simulation_schema},
>>>
>>>
        'Dielectric': {'SimulationLayer': dielectric_reweighting_schema}
>>> }
>>>
>>> property_estimator.request_estimate(
>>>
        data_set,
>>>
        force_field_source,
        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 openff.evaluator.forcefield import ParameterGradientKey
>>> parameter_gradient_keys = [
>>> ParameterGradientKey('Bonds', '[#6:1]-[#8:2]', 'length')
```

(continues on next page)

(continued from previous page)

```
>>> ParameterGradientKey('Angles', '[*:1]-[#8:2]-[*:3]', 'angle')
>>> ]
>>>
property_estimator.request_estimate(
>>> data_set,
>>> force_field_source,
>>> options,
>>> parameter_gradient_keys
>>> )
```

__init__(connection_options=None)

Parameters connection_options (ConnectionOptions, optional) – The options used when connecting to the calculation server. If *None*, default options are used.

Methods

init([connection_options])	param connection_options The options used when connecting to the calcula- tion
<pre>default_request_options(data_set,)</pre>	Returns the default <i>RequestOptions</i> options used to estimate a set of properties if <i>None</i> are provided.
<pre>request_estimate(property_set,[,])</pre>	Submits a request for the <i>EvaluatorServer</i> to attempt to estimate the data set of physical properties using the specified force field parameters according to the provided options.
<pre>retrieve_results(request_id[, synchronous,])</pre>	Retrieves the current results of a request from the server.

Attributes

server_address	The address of the server that this client is connected
	to.
server_port	The port of the server that this client is connected to.

property server_address

The address of the server that this client is connected to.

Type str

property server_port

The port of the server that this client is connected to.

Type int

static default_request_options(data_set, force_field_source)

Returns the default RequestOptions options used to estimate a set of properties if None are provided.

Parameters

- data_set (PhysicalPropertyDataSet) The data set which would be estimated.
- **force_field_source** (ForceFieldSource) The force field parameters which will be used by the request.

Returns The default options.

Return type *RequestOptions*

request_estimate(property_set, force_field_source, options=None, parameter_gradient_keys=None)
Submits a request for the EvaluatorServer to attempt to estimate the data set of physical properties using
the specified force field parameters according to the provided options.

Parameters

- property_set (PhysicalPropertyDataSet) The set of properties to estimate.
- force_field_source (ForceFieldSource or openff.toolkit.typing. engines.smirnoff.ForceField) – The force field parameters to estimate the properties using.
- **options** (RequestOptions, optional) A set of estimator options. If *None* default options will be used (see *default_request_options*).
- **parameter_gradient_keys** (*list of ParameterGradientKey, optional*) A list of the parameters that the physical properties should be differentiated with respect to.

Returns

- Request An object which will provide access to the results of this request.
- *EvaluatorException, optional* Any exceptions raised while attempting the submit the request.

retrieve_results(request_id, synchronous=False, polling_interval=5)

Retrieves the current results of a request from the server.

Parameters

- request_id (str) The server assigned id of the request.
- **synchronous** (*bool*) If true, this method will block the main thread until the server either returns a result or an error.
- **polling_interval** (*float*) If running synchronously, this is the time interval (seconds) between checking if the request has completed.

Returns

- *RequestResult, optional* Returns the current results of the request. This may be *None* if any unexpected exceptions occurred while retrieving the estimate.
- *EvaluatorException, optional* The exception raised will trying to retrieve the result, if any.

BatchMode

class openff.evaluator.client.BatchMode(value)

The different modes in which a server can batch together properties to estimate.

This enum may take values of

- SameComponents: All properties measured for substances containing exactly the same components will be placed into a single batch. E.g. The density of a 80:20 and a 20:80 mix of ethanol and water would be batched together, but the density of pure ethanol and the density of pure water would be placed into separate batches.
- SharedComponents: All properties measured for substances containing at least common component will be batched together. E.g.The densities of 80:20 and 20:80 mixtures of ethanol and water, and the pure densities of ethanol and water would be batched together.

Properties will only be marked as estimated by the server when all properties in a single batch are completed.

___init__()

Attributes

SameComponents

SharedComponents

ConnectionOptions

__init__(server_address=None, server_port=None)

Parameters

- server_address (str) The address of the server to connect to.
- server_port (int) The port of the server to connect to.

Methods

__init__([server_address, server_port])

param server_address The address of the server to connect to.

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

class openff.evaluator.client.ConnectionOptions(server_address=None, server_port=None)
 The options to use when connecting to an EvaluatorServer

Attributes

server_address	The address of the server to connect to.
server_port	The port of the server to connect to.

server_address

The address of the server to connect to. The default value of this attribute is localhost.

Type str

server_port

The port of the server to connect to. The default value of this attribute is 8000.

Type int

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

validate(attribute_type=None)

Validate the values of the attributes. If attribute_type is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

Request

class openff.evaluator.client.Request(client=None)

An estimation request which has been sent to a *EvaluatorServer* instance.

This object can be used to query and retrieve the results of the request when finished, or be stored to retrieve the request at some point in the future.

__init__(client=None)

Parameters client (EvaluatorClient, optional) – The client which submitted this request.

Methods

__init__([client])

param client The client which submitted this request.

<pre>from_j son(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>results([synchronous, polling_interval])</pre>	Attempt to retrieve the results of the request from the
	server.
validate([attribute_type])	Validate the values of the attributes.

Attributes

connection_options	The options used to connect to the server handling
	the request.
id	The unique id assigned to this request by the server.

id

The unique id assigned to this request by the server. The default value of this attribute is not set and must be set by the user.

Type str

connection_options

The options used to connect to the server handling the request. The default value of this attribute is not set and must be set by the user..

Type ConnectionOptions

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

Attempt to retrieve the results of the request from the server.

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

Parameters

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

Returns

- *RequestResult, optional* Returns the current results of the request. This may be *None* if any unexpected exceptions occurred while retrieving the estimate.
- EvaluatorException, optional The exception raised will trying to retrieve the result if any.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

RequestOptions

class openff.evaluator.client.RequestOptions

The options to use when requesting a set of physical properties be estimated by the server.

__init__()

Methods

__init__()

add_schema(layer_type, property_type, schema)	A convenience function for adding a calculation schema to the schema dictionary.
from icon(file math)	Create this object from a JSON file.
<pre>from_json(file_path)</pre>	5
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
validate([attribute_type])	Validate the values of the attributes.

Attributes

batch_mode	The way in which the server should batch together
	properties to estimate.
calculation_layers	The calculation layers which may be used to estimate
	the set of physical properties.
calculation_schemas	The schemas that each calculation layer should use
	when estimating the set of physical properties.

calculation_layers

The calculation layers which may be used to estimate the set of physical properties. The order in which the layers appears in this list determines the order in which the layers will attempt to estimate the data set. The default value of this attribute is ['ReweightingLayer', 'SimulationLayer'].

Type list

calculation_schemas

The schemas that each calculation layer should use when estimating the set of physical properties. The dictionary should be of the form [property_type][layer_type]. The default value of this attribute is not set. This attribute is *optional*.

Type dict

batch_mode

The way in which the server should batch together properties to estimate. Properties will only be marked as finished when all properties in a single batch are completed. The default value of this attribute is BatchMode.SharedComponents. This attribute is *optional*.

Type BatchMode

add_schema(layer_type, property_type, schema)

A convenience function for adding a calculation schema to the schema dictionary.

Parameters

- **layer_type** (*str or type of CalculationLayer*) The layer to associate the schema with.
- **property_type** (*str or type of PhysicalProperty*) The class of property to associate the schema with.
- **schema** (*CalculationSchema*) The schema to add.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

RequestResult

class openff.evaluator.client.RequestResult

The current results of an estimation request - these results may be partial if the server hasn't yet completed the request.

___init__()

Methods

__init__()

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

estimated_properties	The set of properties which have been successfully
	estimated.
exceptions	The set of properties which have yet to be, or are cur-
	rently being estimated.
queued_properties	The set of properties which have yet to be, or are cur-
	rently being estimated.
unsuccessful_properties	The set of properties which could not be successfully
	estimated.

queued_properties

The set of properties which have yet to be, or are currently being estimated.

Type PhysicalPropertyDataSet

estimated_properties

The set of properties which have been successfully estimated.

Type PhysicalPropertyDataSet

unsuccessful_properties

The set of properties which could not be successfully estimated.

Type PhysicalPropertyDataSet

exceptions

The set of properties which have yet to be, or are currently being estimated. The default value of this attribute is [].

Type list

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

Exceptions

EvaluatorException

A serializable wrapper around an *Exception*.

EvaluatorException

exception openff.evaluator.utils.exceptions.**EvaluatorException**(*message=None*) A serializable wrapper around an *Exception*.

classmethod from_exception(exception)

Initialize this class from an existing exception.

Parameters exception (Exception) – The existing exception

Returns The initialized exception object.

Return type cls

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

with_traceback()

Exception.with_traceback(tb) - set self.__traceback__ to tb and return self.

2.32.2 Server Side API

EvaluatorServer	The object responsible for coordinating all properties estimations to be ran using the openff-evaluator frame-work.
Batch	Represents a batch of physical properties which are be- ing estimated by the server for a given set of force field parameters.

EvaluatorServer

enable_data_caching=True, delete_working_files=True)

The object responsible for coordinating all properties estimations to be ran using the openff-evaluator framework.

This server is responsible for receiving estimation requests from the client, determining which calculation layer to use to launch the request, and distributing that estimation across the available compute resources.

Notes

Every client request is split into logical chunk batches. This enables batches of related properties (e.g. all properties for CO) to be estimated in one go (or one task graph in the case of workflow based layers) and returned when ready, rather than waiting for the full data set to complete.

Examples

Setting up a general server instance using a dask based calculation backend, and a local file storage backend:

```
>>> # Create the backend which will be responsible for distributing the calculations
>>> from openff.evaluator.backends.dask import DaskLocalCluster
>>> calculation_backend = DaskLocalCluster()
>>> calculation_backend.start()
>>>
>>> # Create the server to which all estimation requests will be submitted
>>> from openff.evaluator.server import EvaluatorServer
>>> property_server = EvaluatorServer(calculation_backend)
>>>
>>> # Instruct the server to listen for incoming requests
>>> # This command will run until killed.
>>> property_server.start()
```

Parameters

- **calculation_backend** (CalculationBackend) The backend to use for executing calculations.
- **storage_backend** (StorageBackend, optional) The backend to use for storing information from any calculations. If *None*, a default *LocalFileStorage* backend will be used.
- **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.
- **enable_data_caching** (*bool*) Whether the server should attempt to cache any data, mainly the output of simulations, produced by estimation requests for future re-processing (e.g for reweighting).
- **delete_working_files** (*bool*) Whether to delete the working files produced while estimated a batch of properties using a specific calculation layer.

Methods

init(calculation_backend[,])	Constructs a new EvaluatorServer object.
start([asynchronous])	Instructs the server to begin listening for incoming
	requests from any EvaluatorClients.
stop()	Stops the property calculation server and it's pro-
	vided backend.

start(asynchronous=False)

Instructs the server to begin listening for incoming requests from any EvaluatorClients.

Parameters asynchronous (*bool*) – If *True* the server will run on a separate thread in the background, returning control back to the main thread. Otherwise, this function will block the main thread until this server is killed.

stop()

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

Batch

class openff.evaluator.server.Batch

Represents a batch of physical properties which are being estimated by the server for a given set of force field parameters.

The expectation is that this object will be passed between calculation layers, whereby each layer will attempt to estimate each of the *queued_properties*. Those properties which can be estimated will be moved to the *estimated_properties* set, while those that couldn't will remain in the *queued_properties* set ready for the next layer.

__init__()

Methods

___init__()

Create this object from a JSON file.
Returns all attributes of a specific <i>attribute_type</i> .
Creates a JSON representation of this class.
Parses a typed json string into the corresponding class
structure.
Validate the values of the attributes.
-

Attributes

enable_data_caching	Whether the server should attempt to cache any data,
chapic_aaca_caching	mainly the output of simulations, produced by this
	batch for future re-processing (e.g for reweighting).
estimated_properties	The set of properties which have been successfully
	estimated.
exceptions	The set of properties which have yet to be, or are cur-
	rently being estimated.
force_field_id	The id of the force field being used to estimate this
	batch of properties.
id	The unique id of this batch.
options	The options being used to estimate this batch.
parameter_gradient_keys	The parameters that this batch of physical properties
	should be differentiated with respect to.
queued_properties	The set of properties which have yet to be estimated.
unsuccessful_properties	The set of properties which have been could not be
	estimated.

id

The unique id of this batch.

Type str

force_field_id

The id of the force field being used to estimate this batch of properties. The default value of this attribute is not set and must be set by the user.

Type str

options

The options being used to estimate this batch. The default value of this attribute is not set and must be set by the user.

Type RequestOptions

parameter_gradient_keys

The parameters that this batch of physical properties should be differentiated with respect to. The default value of this attribute is not set and must be set by the user.

Type list

enable_data_caching

Whether the server should attempt to cache any data, mainly the output of simulations, produced by this batch for future re-processing (e.g for reweighting). The default value of this attribute is **True**.

Type bool

queued_properties

The set of properties which have yet to be estimated. The default value of this attribute is [].

Type list

estimated_properties

The set of properties which have been successfully estimated. The default value of this attribute is [].

Type list

unsuccessful_properties

The set of properties which have been could not be estimated. The default value of this attribute is [].

Type list

exceptions

The set of properties which have yet to be, or are currently being estimated. The default value of this attribute is [].

Type list

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute type=None) Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

2.32.3 Physical Property API

PhysicalProperty	Represents the value of any physical property and it's
	uncertainty if provided.
PropertyPhase	An enum describing the phase that a property was col-
	lected in.
Source	Container class for information about how a property
	was measured / calculated.
CalculationSource	Contains any metadata about how a physical property
	was calculated.
MeasurementSource	Contains any metadata about how a physical property
	was measured by experiment.

PhysicalProperty

class openff.evaluator.datasets.PhysicalProperty(thermodynamic_state=None,

phase=PropertyPhase.Undefined, substance=None,

value=None, uncertainty=None, source=None)

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

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=PropertyPhase.Undefined, substance=None, value=None, uncertainty=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** (*openff.evaluator.unit.Quantity*) The value of the measured physical property.
- **uncertainty** (*openff.evaluator.unit.Quantity*) The uncertainty in the measured value.
- **source** (Source) The source of this property.

Methods

init([thermodynamic_state, phase,])	Constructs a new PhysicalProperty object.
<pre>default_unit()</pre>	openff.evaluator.unit.Unit: The default unit (e.g.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

gradients	The gradients of this property with respect to differ-
	ent force field parameters.
id	A unique identifier string assigned to this property
metadata	Additional metadata associated with this property.
phase	The phase / phases that this property was measured
	in.
source	The original source of this physical property.
substance	The substance that this property was measured esti-
	mated for.
thermodynamic_state	The thermodynamic state that this propertywas mea-
	sured / estimated at.
uncertainty	The uncertainty in measured / estimated value of this
	property.
value	The measured / estimated value of this property.

abstract classmethod default_unit()

openff.evaluator.unit.Unit: The default unit (e.g. g / mol) associated with this class of property.

id

A unique identifier string assigned to this property

Type str

metadata

Additional metadata associated with this property. All property metadata will be made accessible to estimation workflows. The default value of this attribute is not set. This attribute is *optional*.

Type dict

thermodynamic_state

The thermodynamic state that this property as measured / estimated at. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

phase

The phase / phases that this property was measured in. The default value of this attribute is not set and must be set by the user.

Type PropertyPhase

substance

The substance that this property was measured estimated for. The default value of this attribute is not set and must be set by the user..

Type Substance

value

The measured / estimated value of this property. The default value of this attribute is not set and must be set by the user..

Type Quantity

uncertainty

The uncertainty in measured / estimated value of this property. The default value of this attribute is not set. This attribute is *optional*.

Type Quantity

gradients

The gradients of this property with respect to different force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type list

source

The original source of this physical property. The default value of this attribute is not set. This attribute is *optional*.

Type Source

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

PropertyPhase

class openff.evaluator.datasets.**PropertyPhase**(*value*) An enum describing the phase that a property was collected in.

Examples

Properties measured in multiple phases (e.g. enthalpies of vaporization) can be defined be concatenating *PropertyPhase* enums:

>>> gas_liquid_phase = PropertyPhase.Gas | PropertyPhase.Liquid

__init__()

Methods

from_string(enum_string) Parses a phase enum from its string representation.

Attributes

Undefined

Solid

Liquid

Gas

classmethod from_string(enum_string)

Parses a phase enum from its string representation.

Parameters enum_string (*str*) – The str representation of a *PropertyPhase*

Returns The created enum

Return type *PropertyPhase*

Examples

To round-trip convert a phase enum: >>> phase = PropertyPhase.Liquid | PropertyPhase.Gas >>> phase_str = str(phase) >>> parsed_phase = PropertyPhase.from_string(phase_str)

Source

class openff.evaluator.datasets.Source

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

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

___init__()

Methods

__init__()

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

CalculationSource

```
class openff.evaluator.datasets.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, \ldots) 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

init([fidelity, provenance])	Constructs a new CalculationSource object.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) - The path to load the JSON from.

Returns The parsed class.

Return type cls

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

MeasurementSource

class openff.evaluator.datasets.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

init([doi, reference])	Constructs a new MeasurementSource object.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

Built-in Properties

Density	A class representation of a density property
ExcessMolarVolume	A class representation of an excess molar volume prop-
	erty
DielectricConstant	A class representation of a dielectric property
EnthalpyOfMixing	A class representation of an enthalpy of mixing property
EnthalpyOfVaporization	A class representation of an enthalpy of vaporization
	property
SolvationFreeEnergy	A class representation of a solvation free energy prop-
	erty.
HostGuestBindingAffinity	A class representation of a host-guest binding affinity
	property

Density

class openff.evaluator.properties.Density(thermodynamic_state=None,

phase=PropertyPhase.Undefined, substance=None, value=None, uncertainty=None, source=None)

A class representation of a density property

__init__(thermodynamic_state=None, phase=PropertyPhase.Undefined, substance=None, value=None, uncertainty=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** (*openff.evaluator.unit.Quantity*) The value of the measured physical property.
- **uncertainty** (*openff.evaluator.unit.Quantity*) The uncertainty in the measured value.
- **source** (Source) The source of this property.

Methods

init([thermodynamic_state, phase,])	Constructs a new PhysicalProperty object.
<pre>default_reweighting_schema([])</pre>	Returns the default calculation schema to use when
	estimating this property by reweighting existing data.
<pre>default_simulation_schema([])</pre>	Returns the default calculation schema to use when
	estimating this class of property from direct simula-
	tions.
<pre>default_unit()</pre>	openff.evaluator.unit.Unit: The default unit (e.g.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

gradients	The gradients of this property with respect to differ-
	ent force field parameters.
id	A unique identifier string assigned to this property
metadata	Additional metadata associated with this property.
phase	The phase / phases that this property was measured
	in.
source	The original source of this physical property.
substance	The substance that this property was measured esti-
	mated for.
thermodynamic_state	The thermodynamic state that this propertywas mea-
	sured / estimated at.
uncertainty	The uncertainty in measured / estimated value of this
	property.
value	The measured / estimated value of this property.

classmethod default_unit()

openff.evaluator.unit.Unit: The default unit (e.g. g / mol) associated with this class of property.

static default_simulation_schema(*absolute_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute object>*, *relative_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute object>*, *n_molecules=1000*) \rightarrow

openff.evaluator.layers.simulation.SimulationSchema

Returns the default calculation schema to use when estimating this class of property from direct simulations.

Parameters

- **absolute_tolerance** (*openff.evaluator.unit.Quantity*, *optional*) The absolute tolerance to estimate the property to within.
- **relative_tolerance** (*float*) The tolerance (as a fraction of the properties reported uncertainty) to estimate the property to within.
- **n_molecules** (*int*) The number of molecules to use in the simulation.

Returns The schema to follow when estimating this property.

Return type SimulationSchema

static default_reweighting_schema(absolute_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute</pre>

object>, relative_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute object>, $n_effective_samples=50$) \rightarrow openff.evaluator.layers.reweighting.ReweightingSchema

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

Parameters

- **absolute_tolerance** (*openff.evaluator.unit.Quantity*, *optional*) The absolute tolerance to estimate the property to within.
- **relative_tolerance** (*float*) The tolerance (as a fraction of the properties reported uncertainty) to estimate the property to within.
- **n_effective_samples** (*int*) The minimum number of effective samples to require when reweighting the cached simulation data.

Returns The schema to follow when estimating this property.

Return type ReweightingSchema

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

gradients

The gradients of this property with respect to different force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type list

id

A unique identifier string assigned to this property

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

metadata

Additional metadata associated with this property. All property metadata will be made accessible to estimation workflows. The default value of this attribute is not set. This attribute is *optional*.

Type dict

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

phase

The phase / phases that this property was measured in. The default value of this attribute is not set and must be set by the user..

Type *PropertyPhase*

source

The original source of this physical property. The default value of this attribute is not set. This attribute is *optional*.

Type Source

substance

The substance that this property was measured estimated for. The default value of this attribute is not set and must be set by the user..

Type Substance

thermodynamic_state

The thermodynamic state that this property as measured / estimated at. The default value of this attribute is not set and must be set by the user.

Type ThermodynamicState

uncertainty

The uncertainty in measured / estimated value of this property. The default value of this attribute is not set. This attribute is *optional*.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

value

The measured / estimated value of this property. The default value of this attribute is not set and must be set by the user..

Type Quantity

ExcessMolarVolume

A class representation of an excess molar volume property

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

Methods

<pre>init([thermodynamic_state, phase,])</pre>	Constructs a new PhysicalProperty object.
<pre>default_reweighting_schema([])</pre>	Returns the default calculation schema to use when
	estimating this class of property by re-weighting
	cached simulation data.
<pre>default_simulation_schema([])</pre>	Returns the default calculation schema to use when
	estimating this class of property from direct simula-
	tions.
default_unit()	openff.evaluator.unit.Unit: The default unit (e.g.
<pre>default_unit() from_json(file_path)</pre>	openff.evaluator.unit.Unit: The default unit (e.g. Create this object from a JSON file.
· · · · · · · · · · · · · · · · · · ·	
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_json(file_path) get_attributes([attribute_type])</pre>	Create this object from a JSON file. Returns all attributes of a specific <i>attribute_type</i> .
from_json(file_path)get_attributes([attribute_type])json([file_path, format])	Create this object from a JSON file. Returns all attributes of a specific <i>attribute_type</i> . Creates a JSON representation of this class.
from_json(file_path)get_attributes([attribute_type])json([file_path, format])	Create this object from a JSON file.Returns all attributes of a specific attribute_type.Creates a JSON representation of this class.Parses a typed json string into the corresponding class

Attributes

The gradients of this property with respect to differ-	
ent force field parameters.	
A unique identifier string assigned to this property	
Additional metadata associated with this property.	
The phase / phases that this property was measured	
in.	
The original source of this physical property.	
The substance that this property was measured esti-	
mated for.	
The thermodynamic state that this propertywas mea-	
sured / estimated at.	
The uncertainty in measured / estimated value of this	
property.	
The measured / estimated value of this property.	

classmethod default_unit()

openff.evaluator.unit.Unit: The default unit (e.g. g / mol) associated with this class of property.

< openff.evaluator.attributes.attributes.UndefinedAttribute
object>, n_effective_samples: int = 50) \rightarrow openff.evaluator.layers.reweighting.ReweightingSchema

Returns the default calculation schema to use when estimating this class of property by re-weighting cached simulation data.

Parameters

- absolute_tolerance The absolute tolerance to estimate the property to within.
- **relative_tolerance** The tolerance (as a fraction of the properties reported uncertainty) to estimate the property to within.
- **n_effective_samples** The minimum number of effective samples to require when reweighting the cached simulation data.

Returns

Return type The default re-weighting calculation schema.

classmethod default_simulation_schema(absolute_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute object>, rela-

tive_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute object>, n_molecules=1000) \rightarrow

openff.evaluator.layers.simulation.SimulationSchema

Returns the default calculation schema to use when estimating this class of property from direct simulations.

Parameters

- **absolute_tolerance** (*openff.evaluator.unit.Quantity*, *optional*) The absolute tolerance to estimate the property to within.
- **relative_tolerance** (*float*) The tolerance (as a fraction of the properties reported uncertainty) to estimate the property to within.

• **n_molecules** (*int*) – The number of molecules to use in the simulation.

Returns The schema to follow when estimating this property.

Return type SimulationSchema

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

gradients

The gradients of this property with respect to different force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type list

id

A unique identifier string assigned to this property

Type str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

metadata

Additional metadata associated with this property. All property metadata will be made accessible to estimation workflows. The default value of this attribute is not set. This attribute is *optional*.

Type dict

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

phase

The phase / phases that this property was measured in. The default value of this attribute is not set and must be set by the user.

Type PropertyPhase

source

The original source of this physical property. The default value of this attribute is not set. This attribute is *optional*.

Type Source

substance

The substance that this property was measured estimated for. The default value of this attribute is not set and must be set by the user..

Type Substance

thermodynamic_state

The thermodynamic state that this property as measured / estimated at. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

uncertainty

The uncertainty in measured / estimated value of this property. The default value of this attribute is not set. This attribute is *optional*.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

value

The measured / estimated value of this property. The default value of this attribute is not set and must be set by the user.

Type Quantity

DielectricConstant

class openff.evaluator.properties.DielectricConstant(thermodynamic_state=None,

phase=PropertyPhase.Undefined, substance=None, value=None, uncertainty=None, source=None)

A class representation of a dielectric property

__init__(thermodynamic_state=None, phase=PropertyPhase.Undefined, substance=None, value=None, uncertainty=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** (*openff.evaluator.unit.Quantity*) The value of the measured physical property.

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

Methods

init([thermodynamic_state, phase,])	Constructs a new PhysicalProperty object.
<pre>default_reweighting_schema([])</pre>	Returns the default calculation schema to use when
	estimating this property by reweighting existing data.
<pre>default_simulation_schema([])</pre>	Returns the default calculation schema to use when
	estimating this class of property from direct simula-
	tions.
<pre>default_unit()</pre>	openff.evaluator.unit.Unit: The default unit (e.g.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

gradients	The gradients of this property with respect to differ-	
	ent force field parameters.	
id	A unique identifier string assigned to this property	
metadata	Additional metadata associated with this property.	
phase	The phase / phases that this property was measured	
	in.	
source	The original source of this physical property.	
substance	The substance that this property was measured esti-	
	mated for.	
thermodynamic_state	The thermodynamic state that this propertywas mea-	
	sured / estimated at.	
uncertainty	The uncertainty in measured / estimated value of this	
	property.	
value	The measured / estimated value of this property.	

classmethod default_unit()

openff.evaluator.unit.Unit: The default unit (e.g. g / mol) associated with this class of property.

tive_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute object>, *n_molecules=1000*)

Returns the default calculation schema to use when estimating this class of property from direct simulations.

Parameters

• **absolute_tolerance** (*openff.evaluator.unit.Quantity*, *optional*) – The absolute tolerance to estimate the property to within.

- **relative_tolerance** (*float*) The tolerance (as a fraction of the properties reported uncertainty) to estimate the property to within.
- **n_molecules** (*int*) The number of molecules to use in the simulation.

Returns The schema to follow when estimating this property.

Return type SimulationSchema

static default_reweighting_schema(*absolute_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute* object>, rela-

> *tive_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute object>*, n_effective_samples=50)

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

Parameters

- **absolute_tolerance** (*openff.evaluator.unit.Quantity*, *optional*) The absolute tolerance to estimate the property to within.
- **relative_tolerance** (*float*) The tolerance (as a fraction of the properties reported uncertainty) to estimate the property to within.
- **n_effective_samples** (*int*) The minimum number of effective samples to require when reweighting the cached simulation data.

Returns The schema to follow when estimating this property.

Return type ReweightingSchema

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

gradients

The gradients of this property with respect to different force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type list

id

A unique identifier string assigned to this property

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

• **file_path** (*str*, *optional*) – The (optional) file path to save the JSON file to.

• format (bool) – Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

metadata

Additional metadata associated with this property. All property metadata will be made accessible to estimation workflows. The default value of this attribute is not set. This attribute is *optional*.

Type dict

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

phase

The phase / phases that this property was measured in. The default value of this attribute is not set and must be set by the user.

Type PropertyPhase

source

The original source of this physical property. The default value of this attribute is not set. This attribute is *optional*.

Type Source

substance

The substance that this property was measured estimated for. The default value of this attribute is not set and must be set by the user..

Type Substance

thermodynamic_state

The thermodynamic state that this propertywas measured / estimated at. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

uncertainty

The uncertainty in measured / estimated value of this property. The default value of this attribute is not set. This attribute is *optional*.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

value

The measured / estimated value of this property. The default value of this attribute is not set and must be set by the user.

Type Quantity

EnthalpyOfMixing

class openff.evaluator.properties.EnthalpyOfMixing(thermodynamic_state=None,

phase=PropertyPhase.Undefined, substance=None, value=None, uncertainty=None, source=None)

A class representation of an enthalpy of mixing property

__init__(thermodynamic_state=None, phase=PropertyPhase.Undefined, substance=None, value=None, uncertainty=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** (*openff.evaluator.unit.Quantity*) The value of the measured physical property.
- **uncertainty** (*openff.evaluator.unit.Quantity*) The uncertainty in the measured value.
- **source** (Source) The source of this property.

Methods

Constructs a new PhysicalProperty object.
Returns the default calculation schema to use when
estimating this class of property by re-weighting
cached simulation data.
Returns the default calculation schema to use when
estimating this class of property from direct simula-
tions.
openff.evaluator.unit.Unit: The default unit (e.g.
Create this object from a JSON file.
Returns all attributes of a specific <i>attribute_type</i> .
Creates a JSON representation of this class.
Parses a typed json string into the corresponding class
structure.
Validate the values of the attributes.

Attributes

gradients	The gradients of this property with respect to differ-	
	ent force field parameters.	
id	A unique identifier string assigned to this property	
metadata	Additional metadata associated with this property.	
phase	The phase / phases that this property was measured	
	in.	
source	The original source of this physical property.	
substance	The substance that this property was measured esti-	
	mated for.	
thermodynamic_state	The thermodynamic state that this propertywas mea-	
	sured / estimated at.	
uncertainty	The uncertainty in measured / estimated value of this	
	property.	
value	The measured / estimated value of this property.	

classmethod default_unit()

openff.evaluator.unit.Unit: The default unit (e.g. g / mol) associated with this class of property.

<openff.evaluator.attributes.attributes.UndefinedAttribute object>, n_effective_samples: int = 50) \rightarrow openff.evaluator.layers.reweighting.ReweightingSchema

Returns the default calculation schema to use when estimating this class of property by re-weighting cached simulation data.

Parameters

- absolute_tolerance The absolute tolerance to estimate the property to within.
- **relative_tolerance** The tolerance (as a fraction of the properties reported uncertainty) to estimate the property to within.
- **n_effective_samples** The minimum number of effective samples to require when reweighting the cached simulation data.

Returns

Return type The default re-weighting calculation schema.

classmethod default_simulation_schema(absolute_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute object>, rela-

tive_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute object>, n_molecules=1000) \rightarrow

openff.evaluator.layers.simulation.SimulationSchema

Returns the default calculation schema to use when estimating this class of property from direct simulations.

Parameters

- **absolute_tolerance** (*openff.evaluator.unit.Quantity*, *optional*) The absolute tolerance to estimate the property to within.
- **relative_tolerance** (*float*) The tolerance (as a fraction of the properties reported uncertainty) to estimate the property to within.

• **n_molecules** (*int*) – The number of molecules to use in the simulation.

Returns The schema to follow when estimating this property.

Return type *SimulationSchema*

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

gradients

The gradients of this property with respect to different force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type list

id

A unique identifier string assigned to this property

Type str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

metadata

Additional metadata associated with this property. All property metadata will be made accessible to estimation workflows. The default value of this attribute is not set. This attribute is *optional*.

Type dict

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

phase

The phase / phases that this property was measured in. The default value of this attribute is not set and must be set by the user.

Type PropertyPhase

source

The original source of this physical property. The default value of this attribute is not set. This attribute is *optional*.

Type Source

substance

The substance that this property was measured estimated for. The default value of this attribute is not set and must be set by the user..

Type Substance

thermodynamic_state

The thermodynamic state that this property as measured / estimated at. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

uncertainty

The uncertainty in measured / estimated value of this property. The default value of this attribute is not set. This attribute is *optional*.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

value

The measured / estimated value of this property. The default value of this attribute is not set and must be set by the user.

Type Quantity

EnthalpyOfVaporization

class openff.evaluator.properties.EnthalpyOfVaporization(thermodynamic_state=None,

phase=PropertyPhase.Undefined, substance=None, value=None, uncertainty=None, source=None)

A class representation of an enthalpy of vaporization property

__init__(thermodynamic_state=None, phase=PropertyPhase.Undefined, substance=None, value=None, uncertainty=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** (*openff.evaluator.unit.Quantity*) The value of the measured physical property.

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

Methods

Constructs a new PhysicalProperty object.
Returns the default calculation schema to use when
estimating this property by reweighting existing data.
Returns the default calculation schema to use when
estimating this class of property from direct simula-
tions.
openff.evaluator.unit.Unit: The default unit (e.g.
Create this object from a JSON file.
Returns all attributes of a specific <i>attribute_type</i> .
Creates a JSON representation of this class.
Parses a typed json string into the corresponding class
structure.
Validate the values of the attributes.

Attributes

gradients	The gradients of this property with respect to differ-	
	ent force field parameters.	
id	A unique identifier string assigned to this property	
metadata	Additional metadata associated with this property.	
phase	The phase / phases that this property was measured	
	in.	
source	The original source of this physical property.	
substance	The substance that this property was measured esti-	
	mated for.	
thermodynamic_state	The thermodynamic state that this propertywas mea-	
	sured / estimated at.	
uncertainty	The uncertainty in measured / estimated value of this	
	property.	
value	The measured / estimated value of this property.	

classmethod default_unit()

openff.evaluator.unit.Unit: The default unit (e.g. g / mol) associated with this class of property.

tive_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute object>, *n_molecules=1000*)

Returns the default calculation schema to use when estimating this class of property from direct simulations.

Parameters

• **absolute_tolerance** (*openff.evaluator.unit.Quantity*, *optional*) – The absolute tolerance to estimate the property to within.

- **relative_tolerance** (*float*) The tolerance (as a fraction of the properties reported uncertainty) to estimate the property to within.
- **n_molecules** (*int*) The number of molecules to use in the simulation.

Returns The schema to follow when estimating this property.

Return type SimulationSchema

 ${\tt classmethod \ default_reweighting_schema({\it absolute_tolerance=<} open {\it ff.evaluator.attributes.attributes.UndefinedAttributes.attributes.uttribute$

object>, relative_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute object>, n_effective_samples=50)

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

Parameters

- **absolute_tolerance** (*openff.evaluator.unit.Quantity*, *optional*) The absolute tolerance to estimate the property to within.
- **relative_tolerance** (*float*) The tolerance (as a fraction of the properties reported uncertainty) to estimate the property to within.
- **n_effective_samples** (*int*) The minimum number of effective samples to require when reweighting the cached simulation data.

Returns The schema to follow when estimating this property.

Return type ReweightingSchema

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

gradients

The gradients of this property with respect to different force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type list

id

A unique identifier string assigned to this property

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

• **file_path** (*str*, *optional*) – The (optional) file path to save the JSON file to.

• format (bool) – Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

metadata

Additional metadata associated with this property. All property metadata will be made accessible to estimation workflows. The default value of this attribute is not set. This attribute is *optional*.

Type dict

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

phase

The phase / phases that this property was measured in. The default value of this attribute is not set and must be set by the user.

Type *PropertyPhase*

source

The original source of this physical property. The default value of this attribute is not set. This attribute is *optional*.

Type Source

substance

The substance that this property was measured estimated for. The default value of this attribute is not set and must be set by the user..

Type Substance

thermodynamic_state

The thermodynamic state that this propertywas measured / estimated at. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

uncertainty

The uncertainty in measured / estimated value of this property. The default value of this attribute is not set. This attribute is *optional*.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

value

The measured / estimated value of this property. The default value of this attribute is not set and must be set by the user.

Type Quantity

SolvationFreeEnergy

A class representation of a solvation free energy property.

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

Methods

init([thermodynamic_state, phase,])	Constructs a new PhysicalProperty object.
<pre>default_simulation_schema([])</pre>	Returns the default calculation schema to use when
	estimating this class of property from direct simula-
	tions.
<pre>default_unit()</pre>	openff.evaluator.unit.Unit: The default unit (e.g.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

gradients	The gradients of this property with respect to differ-
-	ent force field parameters.
id	A unique identifier string assigned to this property
metadata	Additional metadata associated with this property.
phase	The phase / phases that this property was measured
	in.
source	The original source of this physical property.
	continues on next page

Table 00 continued non previous page		
substance		The substance that this property was measured esti-
		mated for.
thermodynamic_state		The thermodynamic state that this propertywas mea-
		sured / estimated at.
uncertainty		The uncertainty in measured / estimated value of this
		property.
value		The measured / estimated value of this property.

Table	39 - continued fro	om previous page
-------	--------------------	------------------

classmethod default_unit()

openff.evaluator.unit.Unit: The default unit (e.g. g / mol) associated with this class of property.

tive_tolerance=<openff.evaluator.attributes.attributes.UndefinedAttribute object>, *n_molecules=2000*)

Returns the default calculation schema to use when estimating this class of property from direct simulations.

Parameters

- **absolute_tolerance** (*openff.evaluator.unit.Quantity*, *optional*) The absolute tolerance to estimate the property to within.
- **relative_tolerance** (*float*) The tolerance (as a fraction of the properties reported uncertainty) to estimate the property to within.
- **n_molecules** (*int*) The number of molecules to use in the simulation.

Returns The schema to follow when estimating this property.

Return type SimulationSchema

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

gradients

The gradients of this property with respect to different force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type list

id

A unique identifier string assigned to this property

Type str

json(file_path=None, format=False) Creates a JSON representation of this class. Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

metadata

Additional metadata associated with this property. All property metadata will be made accessible to estimation workflows. The default value of this attribute is not set. This attribute is *optional*.

Type dict

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

phase

The phase / phases that this property was measured in. The default value of this attribute is not set and must be set by the user.

Type *PropertyPhase*

source

The original source of this physical property. The default value of this attribute is not set. This attribute is *optional*.

Type Source

substance

The substance that this property was measured estimated for. The default value of this attribute is not set and must be set by the user..

Type Substance

thermodynamic_state

The thermodynamic state that this propertywas measured / estimated at. The default value of this attribute is not set and must be set by the user.

Type ThermodynamicState

uncertainty

The uncertainty in measured / estimated value of this property. The default value of this attribute is not set. This attribute is *optional*.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

value

The measured / estimated value of this property. The default value of this attribute is not set and must be set by the user..

Type Quantity

HostGuestBindingAffinity

A class representation of a host-guest binding affinity property

- __init__(thermodynamic_state=None, phase=PropertyPhase.Undefined, substance=None, value=None, uncertainty=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** (*openff.evaluator.unit.Quantity*) The value of the measured physical property.
- **uncertainty** (*openff.evaluator.unit.Quantity*) The uncertainty in the measured value.
- **source** (Source) The source of this property.

Methods

Constructs a new PhysicalProperty object.
Returns the default calculation schema to use when
estimating a host-guest binding affinity measurement
with an APR calculation using the paprika package.
openff.evaluator.unit.Unit: The default unit (e.g.
Returns the default calculation schema to use when
estimating this class of property from direct simula-
tions.
Create this object from a JSON file.
Returns all attributes of a specific <i>attribute_type</i> .
Creates a JSON representation of this class.
Parses a typed json string into the corresponding class
structure.
Validate the values of the attributes.

Attributes

gradients	The gradients of this property with respect to differ-
	ent force field parameters.
id	A unique identifier string assigned to this property
metadata	Additional metadata associated with this property.
phase	The phase / phases that this property was measured
	in.
source	The original source of this physical property.
substance	The substance that this property was measured esti-
	mated for.
thermodynamic_state	The thermodynamic state that this propertywas mea-
	sured / estimated at.
uncertainty	The uncertainty in measured / estimated value of this
	property.
value	The measured / estimated value of this property.

classmethod default_unit()

openff.evaluator.unit.Unit: The default unit (e.g. g / mol) associated with this class of property.

static default_yank_schema(existing_schema=None)

Returns the default calculation schema to use when estimating this class of property from direct simulations.

Parameters existing_schema (SimulationSchema, optional) – An existing schema whose settings to use. If set, the schema's *workflow_schema* will be overwritten by this method.

Returns The schema to follow when estimating this property.

Return type SimulationSchema

classmethod default_paprika_schema(existing_schema:

Optional[openff.evaluator.layers.simulation.SimulationSchema] = None, n_solvent_molecules: int = 2500, n_thermalization_steps: int = 50000, n_equilibration_steps: int = 200000, n_production_steps: int = 2500000, dt_thermalization: openff.evaluator.utils.units.Quantity = <Quantity(1.0, 'femtosecond')>, dt_equilibration: openff.evaluator.utils.units.Quantity = <Quantity(2.0, 'femtosecond')>, dt_production: openff.evaluator.utils.units.Quantity = <Quantity(2.0, 'femtosecond')>, debug: bool = False) To use when estimating a host-guest binding affinity measurement

Returns the default calculation schema to use when estimating a host-guest binding affinity measurement with an APR calculation using the paprika package.

Notes

This schema requires additional metadata to be able to estimate each metadata. This metadata is automatically generated for properties loaded from the taproom package using the TaproomDataSet object.

Parameters

- **existing_schema** (SimulationSchema, optional) An existing schema whose settings to use. If set, the schema's *workflow_schema* will be overwritten by this method.
- n_solvent_molecules The number of solvent molecules to add to the box.
- **n_thermalization_steps** The number of thermalization simulations steps to perform. Sample generated during this step will be discarded.
- **n_equilibration_steps** The number of equilibration simulations steps to perform. Sample generated during this step will be discarded.
- **n_production_steps** The number of production simulations steps to perform. Sample generated during this step will be used in the final free energy calculation.
- dt_thermalization The integration timestep during thermalization
- dt_equilibration The integration timestep during equilibration
- dt_production The integration timestep during production
- **debug** Whether to return a debug schema. This is nearly identical to the default schema, albeit with significantly less solvent molecules (10), all simulations run in NVT and much shorter simulation runs (500 steps). If True, the other input arguments will be ignored.

Returns The schema to follow when estimating this property.

Return type SimulationSchema

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

gradients

The gradients of this property with respect to different force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type list

id

A unique identifier string assigned to this property

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

metadata

Additional metadata associated with this property. All property metadata will be made accessible to estimation workflows. The default value of this attribute is not set. This attribute is *optional*.

Type dict

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

phase

The phase / phases that this property was measured in. The default value of this attribute is not set and must be set by the user..

Type *PropertyPhase*

source

The original source of this physical property. The default value of this attribute is not set. This attribute is *optional*.

Type Source

substance

The substance that this property was measured estimated for. The default value of this attribute is not set and must be set by the user..

Type Substance

thermodynamic_state

The thermodynamic state that this propertywas measured / estimated at. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

uncertainty

The uncertainty in measured / estimated value of this property. The default value of this attribute is not set. This attribute is *optional*.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

value

The measured / estimated value of this property. The default value of this attribute is not set and must be set by the user..

Type Quantity

Substance Definition

Substance	Defines the components, their amounts, and their roles
	in a system.
Component	Defines a single component in a chemical system, as well
	as it's role within the system (if any).
Amount	A representation of the amount of a given component in
	a Substance.
ExactAmount	The exact number of instances of a Component in a Sub-
	stance.
MoleFraction	The mole fraction of a <i>Component</i> in a <i>Substance</i> .

Substance

class openff.evaluator.substances.Substance

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

Examples

A neat liquid containing only a single component:

```
>>> from openff.evaluator.substances import Component, ExactAmount, MoleFraction
>>> liquid = Substance()
>>> liquid.add_component(Component(smiles='0'), MoleFraction(1.0))
```

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

```
>>> binary_mixture = Substance()
>>> binary_mixture.add_component(Component(smiles='0'), MoleFraction(0.2))
>>> binary_mixture.add_component(Component(smiles='C0'), 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:

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__()

Methods

__init__()

<pre>add_component(component, amount)</pre>	Add a component to the Substance.
<pre>calculate_aqueous_ionic_mole_fraction()</pre>	Determines what mole fraction of ions is needed to
	yield
<pre>from_components(*components)</pre>	Creates a new Substance object from a list of compo-
	nents.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
get_amounts(component)	Returns the amounts of the component in this sub-
	stance.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_molecules_per_component(maximum_molecules_</pre>	u Returns the number of molecules for each component
	in this substance, given a maximum total number of
	molecules.
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

amounts	the amounts of the component in this substance This attribute is <i>read-only</i> .
components	A list of all of the components in this substance.
identifier	A unique str representation of this substance, which encodes all components and their amounts in the sub-
	stance.
number_of_components	The number of different components in this sub-
	stance.

components

A list of all of the components in this substance. The default value of this attribute is (). This attribute is *read-only*.

Type tuple

amounts

the amounts of the component in this substance This attribute is read-only.

Type dict

property identifier

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

Type str

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 (Component or str) – The components to add to the substance. These may either be full *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** (Component) The component to add to the system.
- **amount** (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* Component) – The component (or it's identifier) to retrieve the amount of.

Returns The amounts of the component in this substance.

Return type tuple of Amount

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.
- **count_exact_amount** (*bool*) Whether components present in an exact amount (i.e. defined with an ExactAmount) should be considered when apply the maximum number

of molecules constraint. This may be set false, for example, when building a separate solvated protein (n = 1) and solvated protein + ligand complex (n = 2) system but wish for both systems to have the same number of solvent molecules.

• **truncate_n_molecules** (*bool*) – Whether or not to attempt to truncate the number of molecules in the substance if the total number is over the specified maximum. If False, an exception will be raised in this case.

The truncation works by iteratively removing one molecule of the predominant component up to a limit of removing a total number of molecules equal to the number of components in the substance (e.g. for a binary substance a maximum of two molecules can be removed). An exception is raised if the number of molecules cannot be sensibly truncated.

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 (*openff.evaluator.unit.Quantity*) – The ionic string in units of molar.

Returns The mole fraction of ions.

Return type float

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

Component

class openff.evaluator.substances.**Component**(*smiles*=<*openff.evaluator.attributes.attributes.UndefinedAttribute object*>, *role*=*Role.Solvent*)

Defines a single component in a chemical system, as well as it's role within the system (if any).

__init__(*smiles=<openff.evaluator.attributes.attributes.UndefinedAttribute object>*, *role=Role.Solvent*) Constructs a new Component object with either a label or a smiles string, but not both.

Notes

The *label* and *smiles* arguments are mutually exclusive, and only one can be passed while the other should be *None*.

Parameters

• smiles (str) - A SMILES descriptor of the component

• role (Component.Role) – The role of this component in the system.

Methods

init([smiles, role])	Constructs a new Component object with either a la-
	bel or a smiles string, but not both.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

identifier	A unique identifier for this component.
role	The role of this component in the system.
smiles	The SMILES pattern which describes this compo-
	nent.

class Role(value)

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 used by workflow to identify the correct species in a system, such as when doing docking or performing solvation free energy calculations.

The SMILES pattern which describes this component. The default value of this attribute is not set and must be set by the user. This attribute is *read-only*.

Type str

role

The role of this component in the system. The default value of this attribute is Role.Solvent. This

smiles

attribute is read-only.

Type Component.Role

property identifier

A unique identifier for this component.

Type str

classmethod from_json(file_path)
 Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

Amount

class openff.evaluator.substances.**Amount**(*value=<openff.evaluator.attributes.attributes.UndefinedAttribute object>*)

A representation of the amount of a given component in a Substance.

__init__(value=<openff.evaluator.attributes.attributes.UndefinedAttribute object>)

Parameters value (float or int) – The value of this amount.

Methods

___init__([value])

param value The value of this amount.

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
get_attributes([attribute_type])	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
to_number_of_molecules(total_substance_molecules)	leconverts this amount to an exact number of
	molecules
validate([attribute_type])	Validate the values of the attributes.

Attributes

identifier	A string identifier for this amount.
value	The value of this amount.

value

The value of this amount. The default value of this attribute is not set and must be set by the user. This attribute is *read-only*.

Type typing.Union[float, int]

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 *to-tal_substance_molecules*.

Return type int

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

validate(attribute_type=None)
Validate the values of the attributes. If attribute_type is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

ExactAmount

 $\verb|class openff.evaluator.substances.ExactAmount(value=< openff.evaluator.attributes.uttributes.UndefinedAttributes)|| \\$

object>)

The exact number of instances of a Component in a Substance.

An assumption is made that this amount is for a component which is infinitely dilute (such as ligands in binding calculations), and hence do not contribute to the total mole fraction of a *Substance*.

__init__(value=<openff.evaluator.attributes.attributes.UndefinedAttribute object>)

Parameters value (float or int) – The value of this amount.

Methods

__init__([value])

param value The value of this amount.

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
to_number_of_molecules(total_substance_molecules	cule©onverts this amount to an exact number of
	molecules
validate([attribute_type])	Validate the values of the attributes.

Attributes

identifier	A string identifier for this amount.
value	The value of this amount.

value

The value of this amount. The default value of this attribute is not set and must be set by the user..

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 *to-tal_substance_molecules*.

Return type int

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

validate(attribute_type=None)

Validate the values of the attributes. If attribute_type is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

MoleFraction

class openff.evaluator.substances.**MoleFraction**(*value=<openff.evaluator.attributes.attributes.UndefinedAttribute object>*)

The mole fraction of a *Component* in a *Substance*.

__init__(value=<openff.evaluator.attributes.attributes.UndefinedAttribute object>)

Parameters value (float or int) – The value of this amount.

Methods

__init__([value])

param value The value of this amount.

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
get_attributes([attribute_type])	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
to_number_of_molecules(total_substance_molecules)	ile©onverts this amount to an exact number of
	molecules
validate([attribute_type])	Validate the values of the attributes.

Attributes

identifier	A string identifier for this amount.
value	The value of this amount.

value

The value of this amount. The default value of this attribute is not set and must be set by the user..

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 *to-tal_substance_molecules*.

Return type int

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(*file_path=None*, *format=False*)

Creates a JSON representation of this class.

Parameters

• **file_path** (*str*, *optional*) – The (optional) file path to save the JSON file to.

• format (bool) – Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

State Definition

ThermodynamicState	Data specifying a physical thermodynamic state obeying
	Boltzmann statistics.

ThermodynamicState

class openff.evaluator.thermodynamics.**ThermodynamicState**(*temperature=None*, *pressure=None*) Data specifying a physical thermodynamic state obeying Boltzmann statistics.

Notes

Equality of two thermodynamic states is determined by comparing the temperature in kelvin to within 3 decimal places, and comparing the pressure (if defined) in pascals to within 3 decimal places.

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 (openff.evaluator.unit.Quantity) The external temperature
- pressure (openff.evaluator.unit.Quantity) The external pressure

Methods

init([temperature, pressure])	Constructs a new ThermodynamicState object.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

beta	Returns one divided by the temperature multiplied by
	the molar gas constant
inverse_beta	Returns the temperature multiplied by the molar gas
	constant
pressure	The external pressure.
temperature	The external temperature.

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

temperature

The external temperature. The default value of this attribute is not set and must be set by the user..

Type Quantity

pressure

The external pressure. The default value of this attribute is not set. This attribute is optional.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

2.32.4 Data Set API

PhysicalPropertyDataSet	An object for storing and curating data sets of both phys-
	ical property measurements and estimated.

PhysicalPropertyDataSet

class openff.evaluator.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

init()	Constructs a new PhysicalPropertyDataSet object.
<pre>add_properties(*physical_properties[, validate])</pre>	Adds a physical property to the data set.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_pandas(data_frame)</pre>	Constructs a data set object from a pandas
	DataFrame object.
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
<pre>merge(data_set[, validate])</pre>	Merge another data set into the current one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>properties_by_substance(substance)</pre>	A generator which may be used to loop over all of
	the properties which were measured for a particular
	substance.
	continues on next page

continues on next page

Table 57 – continued from previous page	
<pre>properties_by_type(property_type)</pre>	A generator which may be used to loop over all of
	properties of a particular type, e.g.
to_pandas()	Converts a PhysicalPropertyDataSet to a pan-
	das.DataFrame object with columns of
validate()	Checks to ensure that all properties within the set are
	valid physical property object.

Table 57 – continued from previous page

Attributes

properties	A list of all of the properties within this set.
property_types	The types of property within this data set.
sources	The sources from which the properties in this data set
	were gathered.
substances	The substances for which the properties in this data
	set were collected for.

property properties

A list of all of the properties within this set.

Type tuple of PhysicalProperty

property property_types

The types of property within this data set.

Type set of str

property substances

The substances for which the properties in this data set were collected for.

Type set of Substance

property sources

The sources from which the properties in this data set were gathered.

Type set of Source

merge(data_set, validate=True)

Merge another data set into the current one.

Parameters

- data_set (PhysicalPropertyDataSet) The secondary data set to merge into this one.
- validate (bool) Whether to validate the other data set before merging.

add_properties(*physical_properties, validate=True)

Adds a physical property to the data set.

Parameters

- physical_properties (PhysicalProperty) The physical property to add.
- validate (bool) Whether to validate the properties before adding them to the set.

properties_by_substance(substance)

A generator which may be used to loop over all of the properties which were measured for a particular substance.

Parameters substance (Substance) – The substance of interest.

Returns

Return type generator of PhysicalProperty

properties_by_type(property_type)

A generator which may be used to loop over all of properties of a particular type, e.g. all "Density" properties.

Parameters property_type (*str or type of PhysicalProperty*) – The type of property of interest. This may either be the string class name of the property or the class type.

Returns

Return type generator of PhysicalProperty

validate()

Checks to ensure that all properties within the set are valid physical property object.

to_pandas()

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

• 'Id'

- 'Temperature (K)'
- 'Pressure (kPa)'
- 'Phase'
- 'N Components'
- 'Component 1'
- 'Role 1'
- 'Mole Fraction 1'
- 'Exact Amount 1'
- ...
- 'Component N'
- 'Role N'
- 'Mole Fraction N'
- 'Exact Amount N'
- '<Property 1> Value (<default unit>)'
- '<Property 1> Uncertainty / (<default unit>)'
- ...
- '<Property N> Value / (<default unit>)'
- '<Property N> Uncertainty / (<default unit>)'
- 'Source'

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

Returns The create data frame.

Return type pandas.DataFrame

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

 $\texttt{classmethod from_pandas}(\textit{data_frame: pandas.core.frame.DataFrame}) \rightarrow \\$

openff.evaluator.datasets.datasets.PhysicalPropertyDataSet

Constructs a data set object from a pandas DataFrame object.

Notes

- All physical properties are assumed to be source from experimental measurements.
- Currently this method onlu supports data frames containing properties which are built-in to the framework (e.g. Density).
- This method assumes the data frame has a structure identical to that produced by the PhysicalPropertyDataSet.to_pandas function.

Parameters data_frame – The data frame to construct the data set from.

Returns

Return type The constructed data set.

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

NIST ThermoML Archive

ThermoMLDataSet	A dataset of physical property measurements created from a ThermoML dataset.		
<pre>register_thermoml_property</pre>	A function used to map a property from the ThermoML archive to an internal <i>PhysicalProperty</i> object of the cor-		
	rect type.		
thermoml_property	A decorator which wraps around the <i>regis-</i> <i>ter_thermoml_property</i> method.		

ThermoMLDataSet

class openff.evaluator.datasets.thermoml.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.		
<pre>add_properties(*physical_properties[, validate])</pre>	Adds a physical property to the data set.		
<pre>from_doi(*doi_list)</pre>	Load a ThermoML data set from a list of DOIs		
<pre>from_file(*file_list)</pre>	Load a ThermoML data set from a list of files		
<pre>from_json(file_path)</pre>	Create this object from a JSON file.		
<pre>from_pandas(data_frame)</pre>	Constructs a data set object from a pandas		
	DataFrame object.		
<pre>from_url(*url_list)</pre>	Load a ThermoML data set from a list of URLs		
<pre>from_xml(xml, default_source)</pre>	Load a ThermoML data set from an xml object.		
json([file_path, format])	Creates a JSON representation of this class.		
<i>merge</i> (data_set[, validate]) Merge another data set into the current			
parse_json(string_contents)	Parses a typed json string into the corresponding class		
	structure.		
<pre>properties_by_substance(substance)</pre>	A generator which may be used to loop over all of		
	the properties which were measured for a particular		
	substance.		
<pre>properties_by_type(property_type)</pre>	A generator which may be used to loop over all of		
	properties of a particular type, e.g.		
to_pandas()	Converts a PhysicalPropertyDataSet to a pan-		
	das.DataFrame object with columns of		
validate()	Checks to ensure that all properties within the set are		
	valid physical property object.		

Attributes

properties	A list of all of the properties within this set.
property_types The types of property within this data set.	
registered_properties	
sources The sources from which the properties in the	

sources	The sources from which the properties in this data set
	were gathered.
substances	The substances for which the properties in this data
	set were collected for.

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*

add_properties(*physical_properties, validate=True)
Adds a physical property to the data set.

Parameters

- physical_properties (PhysicalProperty) The physical property to add.
- validate (bool) Whether to validate the properties before adding them to the set.

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_pandas(*data_frame: pandas.core.frame.DataFrame*) \rightarrow

openff.evaluator.datasets.datasets.PhysicalPropertyDataSet Constructs a data set object from a pandas DataFrame object.

Notes

- All physical properties are assumed to be source from experimental measurements.
- Currently this method onlu supports data frames containing properties which are built-in to the framework (e.g. Density).
- This method assumes the data frame has a structure identical to that produced by the PhysicalPropertyDataSet.to_pandas function.

Parameters data_frame - The data frame to construct the data set from.

Returns

Return type The constructed data set.

classmethod from_xml(xml, default_source)

Load a ThermoML data set from an xml object.

Parameters

- **xml** (*str*) The xml string to parse.
- **default_source** (Source) The source to use if one cannot be parsed from the archive itself.

Returns The loaded ThermoML data set.

Return type ThermoMLDataSet

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(data_set, validate=True)

Merge another data set into the current one.

Parameters

- **data_set** (PhysicalPropertyDataSet) The secondary data set to merge into this one.
- validate (bool) Whether to validate the other data set before merging.

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

property properties

A list of all of the properties within this set.

Type tuple of PhysicalProperty

properties_by_substance(substance)

A generator which may be used to loop over all of the properties which were measured for a particular substance.

Parameters substance (Substance) – The substance of interest.

Returns

Return type generator of PhysicalProperty

properties_by_type(property_type)

A generator which may be used to loop over all of properties of a particular type, e.g. all "Density" properties.

Parameters property_type (*str or type of PhysicalProperty*) – The type of property of interest. This may either be the string class name of the property or the class type.

Returns

Return type generator of PhysicalProperty

property property_types

The types of property within this data set.

Type set of str

property sources

The sources from which the properties in this data set were gathered.

Type set of Source

property substances

The substances for which the properties in this data set were collected for.

Type set of Substance

to_pandas()

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

• 'Id'

- 'Temperature (K)'
- 'Pressure (kPa)'
- 'Phase'
- 'N Components'
- 'Component 1'
- 'Role 1'
- 'Mole Fraction 1'
- 'Exact Amount 1'
- ...
- 'Component N'
- 'Role N'
- 'Mole Fraction N'
- 'Exact Amount N'
- '<Property 1> Value (<default unit>)'

- '<Property 1> Uncertainty / (<default unit>)'
- ...
- '<Property N> Value / (<default unit>)'
- '<Property N> Uncertainty / (<default unit>)'
- 'Source'

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

Returns The create data frame.

Return type pandas.DataFrame

validate()

Checks to ensure that all properties within the set are valid physical property object.

register_thermoml_property

openff.evaluator.datasets.thermoml.register_thermoml_property(*thermoml_string*,

supported_phases,
property_class=None,
conversion_function=None)

A function used to map a property from the ThermoML archive to an internal *PhysicalProperty* object of the correct type.

This function takes either a specific class (e.g. *Density*) which maps directly to the specified *thermoml_string*, or a a function which maps a *ThermoMLProperty* into a *PhysicalProperty* allowing fuller control.

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.
- **property_class** (type of PhysicalProperty, optional) The class associated with this physical property. This argument is mutually exclusive with the *conversion_function* argument.
- **conversion_function** (*function*) A function which maps a *ThermoMLProperty* into a *PhysicalProperty*. This argument is mutually exclusive with the *property_class* argument.

thermoml_property

openff.evaluator.datasets.thermoml.thermoml_property(thermoml_string, supported_phases)
 A decorator which wraps around the register_thermoml_property method.

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.

Taproom

TaproomDataSet	A dataset of host-guest binding affinity measurements
	which sources its data from the taproom package.
TaproomSource	Contains metadata about the source of a host-guest bind-
	ing affinity measurement which was pulled from the
	taproom package.

TaproomDataSet

class openff.evaluator.datasets.taproom.TaproomDataSet(host_codes: Optional[List[str]] = None, guest_codes: Optional[List[str]] = None, default_ionic_strength: Op- tional[openff.evaluator.utils.units.Quantity] = <Quantity(150, 'millimolar')>, negative_buffer_ion: str = '[Cl-]', positive_buffer_ion: str = '[Na+]', attach_apr_meta_data: bool = True)

A dataset of host-guest binding affinity measurements which sources its data from the taproom package.

The loaded HostGuestBindingAffinity properties will also be optionally (enabled by default) initialized with the metadata required by the APR estimation workflow.

- **host_codes** The three letter codes of the host molecules to load from taproom If no list is provided, all hosts will be loaded.
- **guest_codes** The three letter codes of the guest molecules to load from taproom. If no list is provided, all guests will be loaded.
- **default_ionic_strength** The default ionic strength to use for measurements. The value specified in taproom will be ignored and this value used instead. If no value is provided, no buffer will be included.
- **negative_buffer_ion** The SMILES pattern of the negative buffer ion to use. The value specified in taproom will be ignored and this value used instead.
- **positive_buffer_ion** The SMILES pattern of the positive buffer ion to use. The value specified in taproom will be ignored and this value used instead.
- **attach_apr_meta_data** Whether to add the metadata required for an APR based calculation using the paprika based workflow.

Methods

init([host_codes, guest_codes,])			
	param host_codes The three letter codes of the host molecules to load from taproom		
add_properties(*physical_properties[, validate])	Adds a physical property to the data set.		
<pre>from_json(file_path)</pre>	Create this object from a JSON file.		
from_pandas(data_frame)	Constructs a data set object from a pandas DataFrame object.		
json([file_path, format])	Creates a JSON representation of this class.		
<pre>merge(data_set[, validate])</pre>	Merge another data set into the current one.		
<pre>parse_j son(string_contents)</pre>	Parses a typed json string into the corresponding class structure.		
<pre>properties_by_substance(substance)</pre>	A generator which may be used to loop over all of the properties which were measured for a particular substance.		
<pre>properties_by_type(property_type)</pre>	A generator which may be used to loop over all of properties of a particular type, e.g.		
to_pandas()	Converts a <i>PhysicalPropertyDataSet</i> to a <i>pan-</i> <i>das.DataFrame</i> object with columns of		
validate()	Checks to ensure that all properties within the set are valid physical property object.		

Attributes

properties	A list of all of the properties within this set.	
property_types	The types of property within this data set.	
sources	The sources from which the properties in this data set	
	were gathered.	
substances	The substances for which the properties in this data	
	set were collected for.	

add_properties(*physical_properties, validate=True)
Adds a physical property to the data set.

Parameters

- physical_properties (PhysicalProperty) The physical property to add.
- validate (bool) Whether to validate the properties before adding them to the set.

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

Notes

- All physical properties are assumed to be source from experimental measurements.
- Currently this method onlu supports data frames containing properties which are built-in to the framework (e.g. Density).
- This method assumes the data frame has a structure identical to that produced by the PhysicalPropertyDataSet.to_pandas function.

Parameters data_frame – The data frame to construct the data set from.

Returns

Return type The constructed data set.

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(data_set, validate=True)

Merge another data set into the current one.

Parameters

- **data_set** (PhysicalPropertyDataSet) The secondary data set to merge into this one.
- validate (bool) Whether to validate the other data set before merging.

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

```
Parameters string_contents (str or bytes) - The typed json string.
```

Returns The parsed class.

Return type Any

property properties

A list of all of the properties within this set.

Type tuple of PhysicalProperty

properties_by_substance(substance)

A generator which may be used to loop over all of the properties which were measured for a particular substance.

Parameters substance (Substance) – The substance of interest.

Returns

Return type generator of PhysicalProperty

properties_by_type(property_type)

A generator which may be used to loop over all of properties of a particular type, e.g. all "Density" properties.

Parameters property_type (*str or type of PhysicalProperty*) – The type of property of interest. This may either be the string class name of the property or the class type.

Returns

Return type generator of PhysicalProperty

property property_types

The types of property within this data set.

Type set of str

property sources

The sources from which the properties in this data set were gathered.

Type set of Source

property substances

The substances for which the properties in this data set were collected for.

Type set of Substance

to_pandas()

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

- 'Id'
- 'Temperature (K)'
- 'Pressure (kPa)'
- 'Phase'
- 'N Components'
- 'Component 1'
- 'Role 1'
- 'Mole Fraction 1'
- 'Exact Amount 1'
- ...
- 'Component N'
- 'Role N'
- 'Mole Fraction N'
- 'Exact Amount N'
- '<Property 1> Value (<default unit>)'
- '<Property 1> Uncertainty / (<default unit>)'
- ...
- '<Property N> Value / (<default unit>)'
- '<Property N> Uncertainty / (<default unit>)'
- 'Source'

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

Returns The create data frame.

Return type pandas.DataFrame

validate()

Checks to ensure that all properties within the set are valid physical property object.

TaproomSource

```
class openff.evaluator.datasets.taproom.TaproomSource(doi=", comment=", technique=",
```

host_identifier=", guest_identifier=")

Contains metadata about the source of a host-guest binding affinity measurement which was pulled from the taproom package.

__init__(doi=", comment=", technique=", host_identifier=", guest_identifier=") Constructs a new MeasurementSource object.

Parameters

- **doi** (*str*) The DOI for the source
- comment (str) A description of where the value came from in the source.
- technique (str) The technique used to measure this value.
- host_identifier (str) The unique three letter host identifier
- guest_identifier (str) The unique three letter guest identifier

Methods

init([doi, comment, technique,])	Constructs a new MeasurementSource object.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

Data Set Curation

CurationComponent	A base component for curation components which apply a particular operation (such as filtering or data conver- sion) to a data set.
CurationComponentSchema	A base class for schemas which specify how particular curation components should be applied to a data set.

CurationComponent

class openff.evaluator.datasets.curation.components.CurationComponent

A base component for curation components which apply a particular operation (such as filtering or data conversion) to a data set.

__init__()

Methods

init()			

apply()	Apply this curation component to a data set.

classmethod apply(data_set: openff.evaluator.datasets.datasets.PhysicalPropertyDataSet, schema:

openff.evaluator.datasets.curation.components.components.CurationComponentSchema,

 $n_processes: int = 1) \rightarrow openff.evaluator.datasets.datasets.PhysicalPropertyDataSet$ classmethod apply(data_set: pandas.core.frame.DataFrame, schema:

openff.evaluator.datasets.curation.components.components.CurationComponentSchema, $n_{processes: int = 1} \rightarrow \text{pandas.core.frame.DataFrame}$

Apply this curation component to a data set.

Parameters

- **data_set** The data frame to apply the component to.
- **schema** The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

CurationComponentSchema

 $__$ init $__$ (*args: Any, **kwargs: Any) \rightarrow None

Methods

__init__(*args, **kwargs)

CurationWorkflow	A convenience class for applying a set of curation com-
	ponents sequentially to a data set.
CurationWorkflowSchema	A schemas which encodes how a set of curation compo-
	nents should be applied sequentially to a data set.

CurationWorkflow

class openff.evaluator.datasets.curation.workflow.CurationWorkflow

A convenience class for applying a set of curation components sequentially to a data set.

__init__()

Methods

init()	
apply()	Apply each component of this curation workflow to
	an initial data set in sequence.

classmethod apply(*data_set:* openff.evaluator.datasets.datasets.PhysicalPropertyDataSet, *schema:* openff.evaluator.datasets.curation.workflow.CurationWorkflowSchema, *n_processes:*

int = 1) \rightarrow openff.evaluator.datasets.datasets.PhysicalPropertyDataSet

classmethod apply(*data_set: pandas.core.frame.DataFrame*, *schema:*

openff.evaluator.datasets.curation.workflow.CurationWorkflowSchema, $n_{processes}$: int = 1) \rightarrow pandas.core.frame.DataFrame

Apply each component of this curation workflow to an initial data set in sequence.

Parameters

- **data_set** The data set to apply the workflow to. This may either be a data set object or it's pandas representation.
- schema The schema which defines the components to apply.
- **n_processes** The number of processes that each component is allowed to parallelize across.

Returns

Return type The data set which has had the curation workflow applied to it.

CurationWorkflowSchema

Methods

__init__(*args, **kwargs)

Attributes

component_schemas

Filtering

FilterDuplicatesSchema	
FilterDuplicates	A component to remove duplicate data points (within a specified precision) from a data set.
FilterByTemperatureSchema	
FilterByTemperature	A component which will filter out data points which were measured outside of a specified temperature range
FilterByPressureSchema	
FilterByPressure	A component which will filter out data points which were measured outside of a specified pressure range.
FilterByMoleFractionSchema	
FilterByMoleFraction	A component which will filter out data points which were measured outside of a specified mole fraction range.
FilterByRacemicSchema	
FilterByRacemic	A component which will filter out data points which were measured for racemic mixtures.
FilterByElementsSchema	
FilterByElements	A component which will filter out data points which were measured for systems which contain specific ele- ments.
	continues on next page

continues on next page

A component which will apply a filter which only retains properties of specified types.
A component which filters out data points measured for systems whereby the stereochemistry of a number of components is undefined.
A component which filters out data points measured for substances where any of the constituent components have a net non-zero charge.
A component which filters out data points measured for substances which contain or are classed as an ionic liq- uids.
A component which filters the data set so that it only con- tains either a specific set of smiles, or does not contain any of a set of specifically excluded smiles.
A component which filters a data set so that it only con- tains measurements made for molecules which contain (or don't) a set of chemical environments represented by SMIRKS patterns.
A component which filters out data points measured for systems with specified number of components.
A component which filters the data set so that it only contains properties measured for particular substances.
A component which filters a data set so that it only con- tains measurements made for substances which contain specific chemical environments.

FilterDuplicatesSchema

__init__(*args: Any, **kwargs: Any) \rightarrow None

Methods

__init__(*args, **kwargs)

Attributes

mole_fraction_precision

pressure_precision

temperature_precision

type

FilterDuplicates

class openff.evaluator.datasets.curation.components.filtering.**FilterDuplicates** A component to remove duplicate data points (within a specified precision) from a data set.

__init__()

Methods

__init__()

apply(data_set, schema[, n_processes]) Apply this curation component to a data set.

classmethod apply(*data_set*, *schema*, *n_processes=1*) Apply this curation component to a data set.

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FilterByTemperatureSchema

class openff.evaluator.datasets.curation.components.filtering.FilterByTemperatureSchem	a(*args:
	Any, **kwargs: Any)

 $_$ init_(*args: Any, **kwargs: Any) \rightarrow None

Methods

__init__(*args, **kwargs)

Attributes

maximum_temperature

minimum_temperature

type

FilterByTemperature

class openff.evaluator.datasets.curation.components.filtering.**FilterByTemperature** A component which will filter out data points which were measured outside of a specified temperature range

__init__()

Methods

apply(data_set, schema[, n_processes]) Apply this curation component to a data set.

classmethod apply(data_set, schema, n_processes=1)
 Apply this curation component to a data set.

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize

across.

Returns

Return type The data set which has had the component applied to it.

FilterByPressureSchema

 $__init_(*args: Any, **kwargs: Any) \rightarrow None$

Methods

__init__(*args, **kwargs)

Attributes

maximum_pressure

minimum_pressure

type

FilterByPressure

class openff.evaluator.datasets.curation.components.filtering.**FilterByPressure** A component which will filter out data points which were measured outside of a specified pressure range.

__init__()

Methods

___init___()

apply(data_set, schema[, n_processes]) Apply this curation component to a data set.

classmethod apply(data_set, schema, n_processes=1)
 Apply this curation component to a data set.

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.

• **n_processes** – The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FilterByMoleFractionSchema

 $__init_(*args: Any, **kwargs: Any) \rightarrow None$

Methods

__init__(*args, **kwargs)

Attributes

mole_fraction_ranges

type

FilterByMoleFraction

class openff.evaluator.datasets.curation.components.filtering.**FilterByMoleFraction** A component which will filter out data points which were measured outside of a specified mole fraction range.

__init__()

Methods

___init__()

apply(data_set, schema[, n_processes])

Apply this curation component to a data set.

classmethod apply(*data_set*, *schema*, *n_processes=1*) Apply this curation component to a data set.

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.

n_processes – The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FilterByRacemicSchema

class openff.evaluator.datasets.curation.components.filtering.FilterByRacemicSchema(*args:

Any, **kwargs: Any)

 $__init_(*args: Any, **kwargs: Any) \rightarrow None$

Methods

__init__(*args, **kwargs)

Attributes

type

FilterByRacemic

class openff.evaluator.datasets.curation.components.filtering.**FilterByRacemic** A component which will filter out data points which were measured for racemic mixtures.

__init__()

Methods

__init__()

apply(data_set, schema[, n_processes]) Apply this curat

Apply this curation component to a data set.

classmethod apply(data_set, schema, n_processes=1)

Apply this curation component to a data set.

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FilterByElementsSchema

 $_$ init_(**args: Any*, ***kwargs: Any*) \rightarrow None

Methods

___init__(*args, **kwargs)

Attributes

allowed_elements

forbidden_elements

type

FilterByElements

class openff.evaluator.datasets.curation.components.filtering.**FilterByElements** A component which will filter out data points which were measured for systems which contain specific elements.

___init__()

Methods

___init___()

apply(data_set, schema[, n_processes]) Apply this curation component to a data set.

classmethod apply(*data_set*, *schema*, *n_processes=1*) Apply this curation component to a data set.

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.

• **n_processes** – The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FilterByPropertyTypesSchema

 $__init_(*args: Any, **kwargs: Any) \rightarrow None$

Methods

__init__(*args, **kwargs)

Attributes

n_components		
property_types	 	
strict		
type		

FilterByPropertyTypes

class openff.evaluator.datasets.curation.components.filtering.**FilterByPropertyTypes** A component which will apply a filter which only retains properties of specified types.

___init__()

Methods

__init__()

apply(data_set, schema[, n_processes])

Apply this curation component to a data set.

classmethod apply(*data_set*, *schema*, *n_processes=1*) Apply this curation component to a data set.

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FilterByStereochemistrySchema

class openff.evaluator.datasets.curation.components.filtering.FilterByStereochemistrySchema(*args:

Any, **kwargs: Any)

 $_$ init_(**args: Any*, ***kwargs: Any*) \rightarrow None

Methods

__init__(*args, **kwargs)

Attributes

type

FilterByStereochemistry

class openff.evaluator.datasets.curation.components.filtering.FilterByStereochemistry
 A component which filters out data points measured for systems whereby the stereochemistry of a number of
 components is undefined.

__init__()

Methods

___init__()

apply(data_set, schema[, n_processes])

Apply this curation component to a data set.

classmethod apply(*data_set*, *schema*, *n_processes=1*) Apply this curation component to a data set.

Parameters

• **data_set** – The data frame to apply the component to.

- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FilterByChargedSchema

class openff.evaluator.datasets.curation.components.filtering.FilterByChargedSchema(*args:

Any, **kwargs: Any)

 $__init_(*args: Any, **kwargs: Any) \rightarrow None$

Methods

__init__(*args, **kwargs)

Attributes

type

FilterByCharged

class openff.evaluator.datasets.curation.components.filtering.FilterByCharged
 A component which filters out data points measured for substances where any of the constituent components
 have a net non-zero charge.

__init__()

Methods

__init__()

apply(data_set, schema[, n_processes]) Apply this curation component to a data set.

classmethod apply(*data_set*, *schema*, *n_processes=1*) Apply this curation component to a data set.

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.

• **n_processes** – The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FilterBylonicLiquidSchema

 $__init_(*args: Any, **kwargs: Any) \rightarrow None$

Methods

__init__(*args, **kwargs)

Attributes

type

FilterBylonicLiquid

class openff.evaluator.datasets.curation.components.filtering.FilterByIonicLiquid
 A component which filters out data points measured for substances which contain or are classed as an ionic
 liquids.

__init__()

Methods

		~
7 12	 -	()

apply(data_set, schema[, n_processes]) Apply this curation component to a data set.

classmethod apply(*data_set*, *schema*, *n_processes=1*) Apply this curation component to a data set.

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize

across.

Returns

Return type The data set which has had the component applied to it.

FilterBySmilesSchema

class openff.evaluator.datasets.curation.components.filtering.FilterBySmilesSchema(*args:

Any, **kwargs: Any)

__init__(*args: Any, **kwargs: Any) \rightarrow None

Methods

__init__(*args, **kwargs)

Attributes

allow_partial_inclusion

smiles_to_exclude

smiles_to_include

type

FilterBySmiles

class openff.evaluator.datasets.curation.components.filtering.FilterBySmiles

A component which filters the data set so that it only contains either a specific set of smiles, or does not contain any of a set of specifically excluded smiles.

__init__()

Methods

__init__()

apply(data_set, schema[, n_processes])

Apply this curation component to a data set.

classmethod apply(*data_set*, *schema*, *n_processes=1*) Apply this curation component to a data set.

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FilterBySmirksSchema

class openff.evaluator.datasets.curation.components.filtering.FilterBySmirksSchema(*args:

Any, **kwargs: Any)

 $__init_(*args: Any, **kwargs: Any) \rightarrow None$

Methods

__init__(*args, **kwargs)

Attributes

allow_partial_inclusion

smirks_to_exclude

smirks_to_include

type

FilterBySmirks

class openff.evaluator.datasets.curation.components.filtering.FilterBySmirks

A component which filters a data set so that it only contains measurements made for molecules which contain (or don't) a set of chemical environments represented by SMIRKS patterns.

__init__()

Methods

___init__()

apply(data_set, schema[, n_processes]) Apply this curation component to a data set.

classmethod apply(*data_set*, *schema*, *n_processes=1*) Apply this curation component to a data set.

Parameters

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FilterByNComponentsSchema

class openff.evaluator.datasets.curation.components.filtering.FilterByNComponentsSchema(*args:

Any, **kwargs: Any)

 $__init__(*args: Any, **kwargs: Any) \rightarrow None$

Methods

__init__(*args, **kwargs)

Attributes

n_components

type

FilterByNComponents

__init__()

Methods

init()			

apply(data_set, schema[, n_pro	cesses])	Apply this curation component to	o a data set.

classmethod apply(*data_set*, *schema*, *n_processes=1*) Apply this curation component to a data set.

Parameters

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FilterBySubstancesSchema

class openff.evaluator.datasets.curation.components.filtering.FilterBySubstancesSchema(*args:

Any, **kwargs: Any)

 $__init__(*args: Any, **kwargs: Any) \rightarrow None$

Methods

__init__(*args, **kwargs)

Attributes

substances_to_exclude	
<pre>substances_to_include</pre>	
type	

FilterBySubstances

class openff.evaluator.datasets.curation.components.filtering.**FilterBySubstances** A component which filters the data set so that it only contains properties measured for particular substances.

This method is similar to *filter_by_smiles*, however here we explicitly define the full substances compositions, rather than individual smiles which should either be included or excluded.

Examples

To filter the data set to only include measurements for pure methanol, pure benzene or an aqueous ethanol mix:

```
>>> schema = FilterBySubstancesSchema(
>>> substances_to_include=[
>>> ('CO',),
>>> ('C1=CC=CC=C1',),
>>> ('CCO', 'O')
>>> ]
>>> )
```

To filter out measurements made for an aqueous mix of benzene:

```
>>> schema = FilterBySubstancesSchema(
>>> substances_to_exclude=[('0', 'C1=CC=CC=C1')]
>>> )
```

__init__()

Methods

```
___init__()
```

apply(data_set, schema[, n_processes]) Apply this curation component to a data set.

classmethod apply(data_set, schema, n_processes=1)

Apply this curation component to a data set.

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.

• **n_processes** – The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FilterByEnvironmentsSchema

 $__init_(*args: Any, **kwargs: Any) \rightarrow None$

Methods

__*init__*(*args, **kwargs)

Attributes

at_least_one_environment

environments

per_component_environments

strictly_specified_environments

type

FilterByEnvironments

class openff.evaluator.datasets.curation.components.filtering.**FilterByEnvironments** A component which filters a data set so that it only contains measurements made for substances which contain specific chemical environments.

__init__()

Methods

init()			

classmethod apply(*data_set, schema, n_processes=1*) Apply this curation component to a data set.

Parameters

apply(data_set, schema[, n_processes])

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

FreeSolv

ImportFreeSolvSchema

ImportFreeSolv

A component which will import the latest version of the FreeSolv data set from the GitHub repository where it is stored.

Apply this curation component to a data set.

ImportFreeSolvSchema

__init__(*args: Any, **kwargs: Any) \rightarrow None

Methods

__init__(*args, **kwargs)

Attributes

type			

ImportFreeSolv

class openff.evaluator.datasets.curation.components.freesolv.ImportFreeSolv

A component which will import the latest version of the FreeSolv data set from the GitHub repository where it is stored.

__init__()

Methods

__init__()

apply(data_set, schema[, n_processes])

Apply this curation component to a data set.

classmethod apply(data_set, schema, n_processes=1) Apply this curation component to a data set.

Parameters

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

ThermoML

ImportThermoMLDataSchema	
ImportThermoMLData	A component which will import all supported data from the NIST ThermoML archive for (optionally) specified journals.

ImportThermoMLDataSchema

Any)

__init__(**args: Any*, ***kwargs: Any*) \rightarrow None

Methods

__init__(*args, **kwargs)

Attributes

cache_file_name

journal_names

retain_uncertainties

root_archive_url

type

ImportThermoMLData

___init__()

Methods

__init__()

apply(data_set, schema[, n_processes]) Apply this curation component to a data set.

classmethod apply(data_set, schema, n_processes=1)

Apply this curation component to a data set.

Parameters

- **data_set** The data frame to apply the component to.
- **schema** The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

Data Point Selection

SelectSubstancesSchema	
SelectSubstances	A component for selecting a specified number data points which were measured for systems containing a specified set of chemical functionalities.
SelectDataPointsSchema	
SelectDataPoints	A component for selecting a set of data points which are measured as close as possible to a particular set of states.
State	
TargetState	
FingerPrintType	An enumeration.

SelectSubstancesSchema

__init__(**args: Any*, ***kwargs: Any*) \rightarrow None

Methods

__init__(*args, **kwargs)

Attributes

finger_print_type

n_per_environment

per_property

substances_to_exclude

target_environments

type

SelectSubstances

class openff.evaluator.datasets.curation.components.selection.**SelectSubstances** A component for selecting a specified number data points which were measured for systems containing a specified set of chemical functionalities.

__init__()

Methods

__init__()

apply(data_set, schema[, n_processes]) Apply this curation component to a data set.

classmethod apply(*data_set*, *schema*, *n_processes=1*) Apply this curation component to a data set.

Parameters

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

SelectDataPointsSchema

class openff.evaluator.datasets.curation.components.selection.SelectDataPointsSchema(*args:

Any, **kwargs: Any)

 $__init_(*args: Any, **kwargs: Any) \rightarrow None$

Methods

__init__(*args, **kwargs)

Attributes

target_states
type

SelectDataPoints

class openff.evaluator.datasets.curation.components.selection.**SelectDataPoints** A component for selecting a set of data points which are measured as close as possible to a particular set of states.

The points will be chosen so as to try and maximise the number of properties measured at the same condition (e.g. ideally we would have a data point for each property at T=298.15 and p=1atm) as this will maximise the chances that we can extract all properties from a single simulation.

__init__()

Methods

__init__()

apply(data_set, schema[, n_processes]) Apply this curation component to a data set.

classmethod apply(*data_set*, *schema*, *n_processes=1*) Apply this curation component to a data set.

Parameters

- **data_set** The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

State

__init__(**args: Any*, ***kwargs: Any*) \rightarrow None

__init__(*args, **kwargs)

Attributes

mole_fractions

pressure

temperature

TargetState

__init__(**args: Any*, ***kwargs: Any*) \rightarrow None

Methods

__init__(*args, **kwargs)

Attributes

property_types

property_types_validator

states

FingerPrintType

class openff.evaluator.datasets.curation.components.selection.**FingerPrintType**(*value*) An enumeration.

__init__()

Attributes

Tree

MACCS166

Data Conversion

ConvertExcessDensityDataSchema	
ConvertExcessDensityData	A component for converting binary mass density data to excess molar volume data and vice versa where pure density data measured for the components is available.

ConvertExcessDensityDataSchema

__init__(*args: Any, **kwargs: Any) \rightarrow None

Methods

__init__(*args, **kwargs)

Attributes

pressure_precision

temperature_precision

type

ConvertExcessDensityData

class openff.evaluator.datasets.curation.components.conversion.**ConvertExcessDensityData** A component for converting binary mass density data to excess molar volume data and vice versa where pure density data measured for the components is available.

Notes

This protocol may result in duplicate data points being generated. It is recommended to apply the de-duplication filter after this component has been applied.

__init__()

Methods

__init__()

app1y(data_set, schema[, n_processes]) Apply this curation component to a data set.

classmethod apply(data_set, schema, n_processes=1)

Apply this curation component to a data set.

Parameters

- data_set The data frame to apply the component to.
- schema The schema which defines how this component should be applied.
- **n_processes** The number of processes that this component is allowed to parallelize across.

Returns

Return type The data set which has had the component applied to it.

2.32.5 Force Field API

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.
SmirnoffForceFieldSource	A wrapper around force fields based on the SMIRks Na- tive Open Force Field (SMIRNOFF) specification.
TLeapForceFieldSource	A wrapper around Amber force fields which may be applied via the <i>tleap</i> software package.
LigParGenForceFieldSource	A wrapper and the OPLSAAM force field which can be applied via the LigParGen server.

ForceFieldSource

class openff.evaluator.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__()

Methods

__init__()

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

SmirnoffForceFieldSource

class openff.evaluator.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

init([inner_xml])	Constructs a new SmirnoffForceFieldSource object
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_object(force_field)</pre>	Creates a new SmirnoffForceFieldSource from an ex-
	isting ForceField object
<pre>from_path(file_path)</pre>	Creates a new SmirnoffForceFieldSource from the
	file path to a <i>ForceField</i> object.
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>to_force_field()</pre>	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 openff.toolkit.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(*openff.toolkit.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

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

TLeapForceFieldSource

class openff.evaluator.forcefield.TLeapForceFieldSource(leap_source='leaprc.gaff2',

 $cutoff = \langle Quantity(9.0, 'angstrom') \rangle$

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.gaff*2' are supported.
- **cutoff** (*openff.evaluator.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
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.

Attributes

cutoff	The non-bonded interaction cutoff.
leap_source	The parameter file which should be sourced by <i>leap</i>
	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 openff.evaluator.unit.Quantity

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

LigParGenForceFieldSource

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, 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** (*openff.evaluator.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.

init([preferred_charge_model, cutoff,])	Constructs a new LigParGenForceFieldSource object
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
<pre>parse_json(string_contents)</pre>	Parses a typed json string into the corresponding class
	structure.

Attributes

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

class ChargeModel(value)

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 openff.evaluator.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

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

Gradient Estimation

ParameterGradientKey

ParameterGradient

ParameterGradientKey

class openff.evaluator.forcefield.ParameterGradientKey(tag=None, smirks=None, attribute=None)

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

Methods

__init__([tag, smirks, attribute])

Attributes

attribute

smirks

tag

ParameterGradient

class openff.evaluator.forcefield.ParameterGradient(key=None, value=None)

__init__(key=None, value=None)

Methods

__init__([key, value])

Attributes

key

value

2.32.6 Calculation Layers API

CalculationLayer	An abstract representation of a calculation layer whose
ca_ca_ca_ca_ca_ca_ca_ca_ca_ca_ca_ca_ca_c	goal is to estimate a set of physical properties using a sin-
	gle approach, such as a layer which employs direct sim-
	ulations to estimate properties, or one which reweights
	cached simulation data to the same end.
CalculationLayerResult	The result of attempting to estimate a property using a
	CalculationLayer.
CalculationLayerSchema	A schema which encodes the options that a Calculation-
	Layer should use when estimating a given class of phys-
	ical properties.
calculation_layer	A decorator which registers a class as being a calculation
	layer which may be used in property calculations.
<pre>register_calculation_layer</pre>	Registers a class as being a calculation layer which may
	be used in property calculations.
<pre>register_calculation_schema</pre>	Registers the default calculation schema to use when es-
	timating a class of properties (e.g.

CalculationLayer

class openff.evaluator.layers.CalculationLayer

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.

__init__()

___init__()

<pre>required_schema_type()</pre>	Returns the type of CalculationLayerSchema re-
	quired by this layer.
<pre>schedule_calculation(calculation_backend,)</pre>	Submit the proposed calculation to the backend of choice.

abstract classmethod required_schema_type()

Returns the type of CalculationLayerSchema required by this layer.

Returns The required schema type.

Return type type of CalculationLayerSchema

Submit the proposed calculation to the backend of choice.

Parameters

- **calculation_backend** (CalculationBackend) The backend to the submit the calculations to.
- storage_backend (StorageBackend) The backend used to store / retrieve data from previous calculations.
- **layer_directory** (*str*) The directory in which to store all temporary calculation data from this layer.
- **batch** (Batch) The batch of properties to estimate with the layer.
- **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.

CalculationLayerResult

class openff.evaluator.layers.CalculationLayerResult

The result of attempting to estimate a property using a *CalculationLayer*.

__init__()

Methods

___init__()

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.

continues on next page

	Table 157 – continued from previous page
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

data_to_store	Paths to the data objects to store.
exceptions	Any exceptions raised by the layer while estimating
	the property.
physical_property	The estimated property (if the layer was successful).

physical_property

The estimated property (if the layer was successful). The default value of this attribute is not set. This attribute is *optional*.

Type *PhysicalProperty*

data_to_store

Paths to the data objects to store. The default value of this attribute is [].

Type list

exceptions

Any exceptions raised by the layer while estimating the property. The default value of this attribute is [].

Type list

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

CalculationLayerSchema

class openff.evaluator.layers.CalculationLayerSchema

A schema which encodes the options that a *CalculationLayer* should use when estimating a given class of physical properties.

___init___()

Methods

__init__()

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
<pre>parse_json(string_contents)</pre>	Parses a typed json string into the corresponding class
	structure.
validate([attribute_type])	Validate the values of the attributes.

Attributes

absolute_tolerance	The absolute uncertainty that the property should be estimated to within.
relative_tolerance	The relative uncertainty that the property should be estimated to within, i.e <i>relative_tolerance</i> * <i>measured_property.uncertainty</i> .

absolute_tolerance

The absolute uncertainty that the property should be estimated to within. This attribute is mutually exclusive with the *relative_tolerance* attribute. The default value of this attribute is not set. This attribute is *optional*.

Type Quantity

relative_tolerance

The relative uncertainty that the property should be estimated to within, i.e *relative_tolerance* * *measured_property.uncertainty*. This attribute is mutually exclusive with the *absolute_tolerance* attribute. The default value of this attribute is not set. This attribute is *optional*.

Type float

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) - The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

calculation_layer

openff.evaluator.layers.calculation_layer()

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

register_calculation_layer

openff.evaluator.layers.register_calculation_layer(layer_class)

Registers a class as being a calculation layer which may be used in property calculations.

Parameters layer_class (type of CalculationLayer) – The calculation layer to register.

register_calculation_schema

openff.evaluator.layers.register_calculation_schema(property_class, layer_class, schema)
Registers the default calculation schema to use when estimating a class of properties (e.g. Density) with a specific
calculation layer (e.g. the SimulationLayer).

Parameters

- **property_class** (*type of PhysicalProperty*) The class of properties to associate with the specified *calculation_layer* and *property_class*.
- **layer_class** (*type of CalculationLayer*) The calculation layer to associate the schema with.
- **schema** (CalculationLayerSchema or Callable[[CalculationLayerSchema], CalculationLayerSchema]) Either the calculation schema to use, or a function which will create the schema from an existing CalculationLayerSchema.

Built-in Calculation Layers

WorkflowCalculationLayer	An calculation layer which uses the built-in workflow framework to estimate sets of physical properties.
WorkflowCalculationSchema	A schema which encodes the options and the workflow schema that a <i>CalculationLayer</i> should use when es- timating a given class of physical properties using the built-in workflow framework.

WorkflowCalculationLayer

class openff.evaluator.layers.workflow.WorkflowCalculationLayer

An calculation layer which uses the built-in workflow framework to estimate sets of physical properties.

__init__()

Methods

		\sim
1 1 1 1	t	()
	<u> </u>	v

pe of <i>CalculationLayerSchema</i> re- yer.
posed calculation to the backend of
of <i>WorkflowResult</i> to a list of <i>Calcu-</i> <i>ult</i> objects.

static workflow_to_layer_result(*queued_properties*, *provenance*, *workflow_results*, **_) Converts a list of *WorkflowResult* to a list of *CalculationLayerResult* objects.

Parameters

- **queued_properties** (*list of PhysicalProperty*) The properties being estimated by this layer
- provenance (dict of str and str) The provenance of each property.
- workflow_results (list of WorkflowResult) The results of each workflow.

Returns The calculation layer result objects.

Return type list of CalculationLayerResult

abstract classmethod required_schema_type()

Returns the type of CalculationLayerSchema required by this layer.

Returns The required schema type.

Return type type of CalculationLayerSchema

Submit the proposed calculation to the backend of choice.

Parameters

- **calculation_backend** (CalculationBackend) The backend to the submit the calculations to.
- **storage_backend** (StorageBackend) The backend used to store / retrieve data from previous calculations.
- **layer_directory** (*str*) The directory in which to store all temporary calculation data from this layer.
- **batch** (Batch) The batch of properties to estimate with the layer.
- **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.

WorkflowCalculationSchema

class openff.evaluator.layers.workflow.WorkflowCalculationSchema

A schema which encodes the options and the workflow schema that a *CalculationLayer* should use when estimating a given class of physical properties using the built-in workflow framework.

__init__()

___init__()

<i>from_json</i> (file_path)	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
validate([attribute_type])	Validate the values of the attributes.

Attributes

absolute_tolerance	The absolute uncertainty that the property should be estimated to within.
relative_tolerance	The relative uncertainty that the property should be estimated to within, i.e <i>relative_tolerance</i> * <i>measured_property.uncertainty</i> .
workflow_schema	The workflow schema to use when estimating properties.

workflow_schema

The workflow schema to use when estimating properties. The default value of this attribute is not set and must be set by the user.

Type WorkflowSchema

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

absolute_tolerance

The absolute uncertainty that the property should be estimated to within. This attribute is mutually exclusive with the *relative_tolerance* attribute. The default value of this attribute is not set. This attribute is *optional*.

Type Quantity

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

relative_tolerance

The relative uncertainty that the property should be estimated to within, i.e *relative_tolerance* * *measured_property.uncertainty*. This attribute is mutually exclusive with the *absolute_tolerance* attribute. The default value of this attribute is not set. This attribute is *optional*.

Type float

SimulationLayer	A calculation layer which employs molecular simulation
	to estimate sets of physical properties.
SimulationSchema	A schema which encodes the options and the workflow
	schema that the SimulationLayer should use when es-
	timating a given class of physical properties using the
	built-in workflow framework.

SimulationLayer

class openff.evaluator.layers.simulation.SimulationLayer

A calculation layer which employs molecular simulation to estimate sets of physical properties.

__init__()

Methods

___init__()

<pre>required_schema_type()</pre>	Returns the type of <i>CalculationLayerSchema</i> required by this layer.
<pre>schedule_calculation(calculation_backend,)</pre>	Submit the proposed calculation to the backend of choice.
<pre>workflow_to_layer_result(queued_properties,)</pre>	Converts a list of <i>WorkflowResult</i> to a list of <i>CalculationLayerResult</i> objects.

classmethod required_schema_type()

Returns the type of CalculationLayerSchema required by this layer.

Returns The required schema type.

Return type type of CalculationLayerSchema

Submit the proposed calculation to the backend of choice.

Parameters

- **calculation_backend** (CalculationBackend) The backend to the submit the calculations to.
- **storage_backend** (StorageBackend) The backend used to store / retrieve data from previous calculations.
- **layer_directory** (*str*) The directory in which to store all temporary calculation data from this layer.
- **batch** (Batch) The batch of properties to estimate with the layer.
- **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.

static workflow_to_layer_result(queued_properties, provenance, workflow_results, **_)
Converts a list of WorkflowResult to a list of CalculationLayerResult objects.

Parameters

- **queued_properties** (*list of PhysicalProperty*) The properties being estimated by this layer
- provenance (dict of str and str) The provenance of each property.
- workflow_results (list of WorkflowResult) The results of each workflow.

Returns The calculation layer result objects.

Return type list of CalculationLayerResult

SimulationSchema

class openff.evaluator.layers.simulation.SimulationSchema

A schema which encodes the options and the workflow schema that the *SimulationLayer* should use when estimating a given class of physical properties using the built-in workflow framework.

__init__()

___init__()

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
validate([attribute_type])	Validate the values of the attributes.

Attributes

absolute_tolerance	The absolute uncertainty that the property should be estimated to within.
relative_tolerance	The relative uncertainty that the property should be estimated to within, i.e <i>relative_tolerance</i> * <i>measured_property.uncertainty</i> .
workflow_schema	The workflow schema to use when estimating prop- erties.

absolute_tolerance

The absolute uncertainty that the property should be estimated to within. This attribute is mutually exclusive with the *relative_tolerance* attribute. The default value of this attribute is not set. This attribute is *optional*.

Type Quantity

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

```
Parameters string_contents (str or bytes) – The typed json string.
```

Returns The parsed class.

Return type Any

relative_tolerance

The relative uncertainty that the property should be estimated to within, i.e *relative_tolerance* * *measured_property.uncertainty*. This attribute is mutually exclusive with the *absolute_tolerance* attribute. The default value of this attribute is not set. This attribute is *optional*.

Type float

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

workflow_schema

The workflow schema to use when estimating properties. The default value of this attribute is not set and must be set by the user.

Type WorkflowSchema

ReweightingLayer	A <i>CalculationLayer</i> which attempts to 'reweight' cached simulation data to evaluate the values of properties at states which have not previously been simulated directly, but where simulations at similar states have been run previously.
ReweightingSchema	A schema which encodes the options and the workflow schema that the <i>SimulationLayer</i> should use when es- timating a given class of physical properties using the built-in workflow framework.
default_storage_query	Return the default query to use when retrieving cached simulation

ReweightingLayer

class openff.evaluator.layers.reweighting.ReweightingLayer

A *CalculationLayer* which attempts to 'reweight' cached simulation data to evaluate the values of properties at states which have not previously been simulated directly, but where simulations at similar states have been run previously.

__init__()

__init__()

<pre>required_schema_type()</pre>	Returns the type of <i>CalculationLayerSchema</i> required by this layer.
<pre>schedule_calculation(calculation_backend,)</pre>	Submit the proposed calculation to the backend of choice.
<pre>workflow_to_layer_result(queued_properties,)</pre>	Converts a list of <i>WorkflowResult</i> to a list of <i>CalculationLayerResult</i> objects.

classmethod required_schema_type()

Returns the type of CalculationLayerSchema required by this layer.

Returns The required schema type.

Return type type of CalculationLayerSchema

classmethod schedule_calculation(calculation_backend, storage_backend, layer_directory, batch, callback, synchronous=False)

Submit the proposed calculation to the backend of choice.

Parameters

- **calculation_backend** (CalculationBackend) The backend to the submit the calculations to.
- **storage_backend** (StorageBackend) The backend used to store / retrieve data from previous calculations.
- **layer_directory** (*str*) The directory in which to store all temporary calculation data from this layer.
- **batch** (Batch) The batch of properties to estimate with the layer.
- **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.

static workflow_to_layer_result(queued_properties, provenance, workflow_results, **_)
Converts a list of WorkflowResult to a list of CalculationLayerResult objects.

Parameters

- **queued_properties** (*list of PhysicalProperty*) The properties being estimated by this layer
- provenance (dict of str and str) The provenance of each property.
- workflow_results (list of WorkflowResult) The results of each workflow.

Returns The calculation layer result objects.

Return type list of CalculationLayerResult

ReweightingSchema

class openff.evaluator.layers.reweighting.ReweightingSchema

A schema which encodes the options and the workflow schema that the *SimulationLayer* should use when estimating a given class of physical properties using the built-in workflow framework.

__init__()

Methods

__init__()

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
validate([attribute_type])	Validate the values of the attributes.

Attributes

absolute_tolerance	The absolute uncertainty that the property should be
	estimated to within.
maximum_data_points	The maximum number of data points to include as
	part of the multi-state reweighting calculations.
relative_tolerance	The relative uncertainty that the property should be
	estimated to within, i.e relative_tolerance * mea-
	sured_property.uncertainty.
storage_queries	The queries to perform when retrieving data for each
	of the components in the system from the storage
	backend.
temperature_cutoff	The maximum difference between the target temper-
	ature and the temperature at which cached data was
	collected to.
workflow_schema	The workflow schema to use when estimating prop-
	erties.

storage_queries

The queries to perform when retrieving data for each of the components in the system from the storage backend. The keys of this dictionary will correspond to the metadata keys made available to the workflow system.

Type dict

maximum_data_points

The maximum number of data points to include as part of the multi-state reweighting calculations. If zero, no cap will be applied. The default value of this attribute is 4.

Type int

temperature_cutoff

The maximum difference between the target temperature and the temperature at which cached data was

collected to. Data collected for temperatures outside of this cutoff will be ignored. The default value of this attribute is 5.0 K.

Type Quantity

validate(attribute type=None)

Validate the values of the attributes. If *attribute type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

absolute_tolerance

The absolute uncertainty that the property should be estimated to within. This attribute is mutually exclusive with the *relative_tolerance* attribute. The default value of this attribute is not set. This attribute is optional.

Type Quantity

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute type=None) Returns all attributes of a specific attribute_type.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

relative_tolerance

The relative uncertainty that the property should be estimated to within, i.e relative_tolerance * measured_property.uncertainty. This attribute is mutually exclusive with the absolute_tolerance attribute. The default value of this attribute is not set. This attribute is optional.

Type float

workflow_schema

The workflow schema to use when estimating properties. The default value of this attribute is not set and must be set by the user.

Type WorkflowSchema

default_storage_query

```
openff.evaluator.layers.reweighting.default_storage_query()
```

Return the default query to use when retrieving cached simulation data from the storage backend.

Currently this query will search for data for the full substance of interest in the liquid phase.

Returns A single query with a key of "*full_system_data*".

Return type dict of str and SimulationDataQuery

2.32.7 Calculation Backends API

CalculationBackend	An abstract base representation of an openff-evaluator calculation backend.
ComputeResources	An object which stores how many of each type of com- putational resource (threads or gpu's) is available to a calculation worker.
QueueWorkerResources	An extended resource object with properties specific to calculations which will run on queue based resources, such as LSF, PBS or SLURM.

CalculationBackend

class openff.evaluator.backends.**CalculationBackend**(*number_of_workers=1*,

resources_per_worker=None)

An abstract base representation of an openff-evaluator calculation 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=None)
Constructs a new CalculationBackend 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, *optional*) The number of resources to request per worker.

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

Attributes

started Returns whether this backend has been started yet.
--

property started

Returns whether this backend has been started yet.

Type bool

start()

Start the calculation backend.

abstract stop()

Stop the calculation backend.

abstract 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 openff.evaluator.backends.ComputeResources(number_of_threads=1, number_of_gpus=0,
```

preferred_gpu_toolkit=GPUToolkit.CUDA)

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

__init__(number_of_threads=1, number_of_gpus=0, preferred_gpu_toolkit=GPUToolkit.CUDA) 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.

<pre>init([number_of_threads,])</pre>	Constructs a new ComputeResources object.

Attributes

gpu_device_indices	The indices of the GPUs to run on.
51	
number_of_gpus	The number of GPUs available to a calculation
	worker.
number_of_threads	The number of threads available to a calculation
	worker.
preferred_gpu_toolkit	The preferred toolkit to use when running on GPUs.

class GPUToolkit(value)

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 openff.evaluator.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.

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

init([number_of_threads,]) Constructs a new ComputeResources object.

Attributes

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

Type simtk.Quantity

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 *HH:MM* where HH is the number of hours and MM the number of minutes

Type str

class GPUToolkit(value)

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 str

property number_of_gpus

The number of GPUs available to a calculation worker.

Type int

property number_of_threads

The number of threads available to a calculation worker.

Type int

property preferred_gpu_toolkit

The preferred toolkit to use when running on GPUs.

Type ComputeResources.GPUToolkit

Dask Backends

BaseDaskBackend	A base <i>dask</i> backend class, which implements function- ality which is common to all other <i>dask</i> based backends.
BaseDaskJobQueueBackend	An openff-evaluator backend which uses a
	<i>dask_jobqueue.JobQueueCluster</i> object to run cal- culations within an existing HPC queuing system.
DaskLocalCluster	An openff-evaluator backend which uses a dask Local-
	<i>Cluster</i> object to run calculations on a single machine.
DaskLSFBackend	An openff-evaluator backend which uses a dask_jobqueue.LSFCluster object to run calcula-
De al-DDCDe al-and	tions within an existing LSF queue.
DaskPBSBackend	An openff-evaluator backend which uses a <i>dask_jobqueue.PBSCluster</i> object to run calcula-tions within an existing PBS queue.

BaseDaskBackend

class openff.evaluator.backends.dask.BaseDaskBackend(number_of_workers=1, re-

sources_per_worker=<openff.evaluator.backends.backends.Comp 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=<openff.evaluator.backends.backends.ComputeResources object>) Constructs a new BaseDaskBackend object.

Methods

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

Attributes

started Returns whether this backend has been started yet.

start()

Start the calculation backend.

stop()

Stop the calculation backend.

property started

Returns whether this backend has been started yet.

Type bool

abstract 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

BaseDaskJobQueueBackend

class openff.evaluator.backends.dask.BaseDaskJobQueueBackend(minimum_number_of_workers=1,

maximum_number_of_workers=1,

re-

sources_per_worker=<openff.evaluator.backends.backen

An openff-evaluator backend which uses a *dask_jobqueue.JobQueueCluster* object to run calculations within an existing HPC queuing system.

See also:

dask_jobqueue.JobQueueCluster

__init__(minimum_number_of_workers=1, maximum_number_of_workers=1,

resources_per_worker=<openff.evaluator.backends.backends.QueueWorkerResources object>, queue_name='default', setup_script_commands=None, extra_script_options=None, adaptive_interval='10000ms', disable_nanny_process=False, cluster_type=None, adaptive_class=None)

Constructs a new BaseDaskJobQueueBackend object

Parameters

- **minimum_number_of_workers** (*int*) The minimum number of workers to request from the queue system.
- **maximum_number_of_workers** (*int*) The maximum number of workers to request from the queue system.

- **resources_per_worker** (QueueWorkerResources) The resources to request per worker.
- queue_name (str) The name of the queue which the workers will be requested from.
- **setup_script_commands** (*list of str*) A list of bash script commands to call within the queue submission script before the call to launch the dask worker.

This may include activating a python environment, or loading an environment module

• **extra_script_options** (*list of str*) – A list of extra job specific options to include in the queue submission script. These will get added to the script header in the form

#BSUB <extra_script_options[x]>

- **adaptive_interval** (*str*) The interval between attempting to either scale up or down the cluster, of of the from 'XXXms'.
- **disable_nanny_process** (*bool*) If true, dask workers will be started in *-no-nanny* mode. This is required if using multiprocessing code within submitted tasks.

This has not been fully tested yet and my lead to stability issues with the workers.

• **adaptive_class** (class of type *distributed.deploy.AdaptiveCore*, optional) – An optional class to pass to dask to use for its adaptive scaling handling. This is mainly exposed to allow easily working around certain dask bugs / quirks.

Methods

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

Attributes

started Returns whether this backend has been started yet.
Started Returns whether this backend has been started yet.

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.

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

property started

Returns whether this backend has been started yet.

Type bool

stop()

Stop the calculation backend.

DaskLocalCluster

class openff.evaluator.backends.dask.DaskLocalCluster(number_of_workers=1, re-

sources_per_worker=<openff.evaluator.backends.backends.Com

object>)

An openff-evaluator backend which uses a *dask LocalCluster* object to run calculations on a single machine.

See also:

dask.LocalCluster

Methods

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

Attributes

started

Returns whether this backend has been started yet.

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

property started

Returns whether this backend has been started yet.

Type bool

stop()

Stop the calculation backend.

DaskLSFBackend

class openff.evaluator.backends.dask.DaskLSFBackend(minimum_number_of_workers=1,

maximum_number_of_workers=1, resources_per_worker=<openff.evaluator.backends.backends.Queue object>, queue_name='default', setup_script_commands=None, extra_script_options=None, adaptive_interval='10000ms', disable_nanny_process=False, adaptive_class=None)

An openff-evaluator backend which uses a *dask_jobqueue.LSFCluster* object to run calculations within an existing LSF queue.

See also:

dask_jobqueue.LSFCluster, DaskPBSBackend

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 openff.evaluator.backends import QueueWorkerResources
>>>
>>> resources = QueueWorkerResources(number_of_threads=1,
>>>
                                      number_of_gpus=1,
                                       preferred_gpu_toolkit=QueueWorkerResources.
>>>
\rightarrow GPUToolkit.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',
>>> 1
>>>
>>> # Define extra options to only run on certain node groups
>>> 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.
\rightarrow load.
```

(continues on next page)

(continued from previous page)

>>>	>> from openff.evaluator.backends.dask import DaskLSFBackend		
>>>			
>>>	<pre>>>> lsf_backend = DaskLSFBackend(minimum_number_of_workers=1,</pre>		
>>>	maximum_number_of_workers=10,		
>>>	<pre>resources_per_worker=resources,</pre>		
>>>	queue_name= <mark>'gpuqueue</mark> ',		
>>>	<pre>setup_script_commands=setup_script_commands,</pre>		
>>>	<pre>extra_script_options=extra_script_options)</pre>		

Methods

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

Attributes

started	Returns whether this backend has been started yet.

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.

property started

Returns whether this backend has been started yet.

Type bool

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

DaskPBSBackend

class openff.evaluator.backends.dask.DaskPBSBackend(minimum_number_of_workers=1,

maximum_number_of_workers=1, resources_per_worker=<openff.evaluator.backends.backends.Queue object>, queue_name='default', setup_script_commands=None, extra_script_options=None, adaptive_interval='10000ms', disable_nanny_process=False, resource_line=None, adaptive_class=None)

An openff-evaluator backend which uses a *dask_jobqueue.PBSCluster* object to run calculations within an existing PBS queue.

See also:

dask_jobqueue.LSFCluster, DaskLSFBackend

Parameters resource_line (*str*) – The string to pass to the *#PBS* -*l* line.

Examples

To create a PBS 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 openff.evaluator.backends import QueueWorkerResources
>>>
>>> resources = QueueWorkerResources(number_of_threads=1,
>>>
                                      number_of_gpus=1,
                                      preferred_gpu_toolkit=QueueWorkerResources.
>>>
\rightarrow GPUToolkit.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',
>>> 1
>>>
>>> # Create the backend which will adaptively try to spin up between one and
>>> # ten workers with the requested resources depending on the calculation.
\rightarrow load.
>>> from openff.evaluator.backends.dask import DaskPBSBackend
>>>
>>> pbs_backend = DaskPBSBackend(minimum_number_of_workers=1,
                                  maximum_number_of_workers=10,
>>>
```

(continues on next page)

(continued from previous page)

>>>	resources_per_worker=resources,
>>>	queue_name='gpuqueue',
>>>	<pre>setup_script_commands=setup_script_commands)</pre>

Methods

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

Attributes

started yet.

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.

property started

Returns whether this backend has been started yet.

Type bool

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

2.32.8 Storage API

StorageBackend	An abstract base representation of how the openff-
	evaluator will interact with and store simulation data.

StorageBackend

class openff.evaluator.storage.StorageBackend

An abstract base representation of how the openff-evaluator will interact with and store simulation data.

Notes

When implementing this class, only private methods should be overridden as the public methods only mainly implement thread locks, while their private version perform their actual function.

__init__()

Constructs a new StorageBackend object.

Methods

init()	Constructs a new StorageBackend object.
has_force_field(force_field)	A convenience method for checking whether the
	specified ForceFieldSource object is stored in the
	backend.
has_object(storage_object)	Checks whether a given hashable object exists in the
	storage system.
query(data_query)	Query the storage backend for data matching the
	query criteria.
<pre>retrieve_force_field(storage_key)</pre>	A convenience method for retrieving ForceField-
	Source objects.
<pre>retrieve_object(storage_key[, expected_type])</pre>	Retrieves a stored object for the estimators storage
	system.
<pre>store_field(force_field)</pre>	A convenience method for storing ForceFieldSource
	objects.
<pre>store_object(object_to_store[,])</pre>	Store an object in the storage system, returning the
	key of the stored object.

store_object(object_to_store, ancillary_data_path=None)

Store an object in the storage system, returning the key of the stored object. This may be different to *storage_key* depending on whether the same or a similar object was already present in the system.

Parameters

- object_to_store (BaseStoredData) The object to store.
- **ancillary_data_path** (*str*, *optional*) The data path to the ancillary directory-like data to store alongside the object if the data type requires one.

Returns The unique key assigned to the stored object.

Return type str

store_field(force_field)

A convenience method for storing ForceFieldSource objects.

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

Returns The unique id of the stored force field.

Return type str

retrieve_object(storage_key, expected_type=None)

Retrieves a stored object for the estimators storage system.

Parameters

- **storage_key** (*str*) A unique key that describes where the stored object can be found within the storage system.
- **expected_type** (*type of BaseStoredData, optional*) The expected data type. An exception is raised if the retrieved data doesn't match the type.

Returns

- BaseStoredData, optional The stored object if the object key is found, otherwise None.
- *str*, *optional* The path to the ancillary data if present.

retrieve_force_field(storage_key)

A convenience method for retrieving *ForceFieldSource* objects.

Parameters storage_key (str) – The key of the force field to retrieve.

Returns The retrieved force field source.

Return type ForceFieldSource

has_object(storage_object)

Checks whether a given hashable object exists in the storage system.

Parameters storage_object (BaseStoredData) - The object to check for.

Returns The unique key of the object if it is in the system, None otherwise.

Return type str, optional

has_force_field(force_field)

A convenience method for checking whether the specified ForceFieldSource object is stored in the backend.

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

Returns The unique key of the object if it is in the system, None otherwise.

Return type str, optional

query(data_query)

Query the storage backend for data matching the query criteria.

Parameters data_query (BaseDataQuery) – The query to perform.

Returns The data that matches the query partitioned by the matched values. The list values take the form (storage_key, data_object, data_directory_path).

Return type dict of tuple and list of tuple of str, BaseStoredData and str

Built-in Storage Backends

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

LocalFileStorage

class openff.evaluator.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 StorageBackend object.

Methods

<i>init</i> ([root_directory])	Constructs a new StorageBackend object.
<pre>has_force_field(force_field)</pre>	A convenience method for checking whether the
	specified ForceFieldSource object is stored in the
	backend.
<pre>has_object(storage_object)</pre>	Checks whether a given hashable object exists in the
	storage system.
<i>query</i> (data_query)	Query the storage backend for data matching the
	query criteria.
<pre>retrieve_force_field(storage_key)</pre>	A convenience method for retrieving ForceField-
	Source objects.
<pre>retrieve_object(storage_key[, expected_type])</pre>	Retrieves a stored object for the estimators storage
	system.
<pre>store_force_field(force_field)</pre>	A convenience method for storing <i>ForceFieldSource</i>
	objects.
<pre>store_object(object_to_store[,])</pre>	Store an object in the storage system, returning the
	key of the stored object.

Attributes

root_directory	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

has_force_field(force_field)

A convenience method for checking whether the specified *ForceFieldSource* object is stored in the backend.

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

Returns The unique key of the object if it is in the system, *None* otherwise.

Return type str, optional

has_object(storage_object)

Checks whether a given hashable object exists in the storage system.

Parameters storage_object (BaseStoredData) - The object to check for.

Returns The unique key of the object if it is in the system, *None* otherwise.

Return type str, optional

query(data_query)

Query the storage backend for data matching the query criteria.

Parameters data_query (BaseDataQuery) – The query to perform.

Returns The data that matches the query partitioned by the matched values. The list values take the form (storage_key, data_object, data_directory_path).

Return type dict of tuple and list of tuple of str, BaseStoredData and str

retrieve_force_field(storage_key)

A convenience method for retrieving ForceFieldSource objects.

Parameters storage_key (str) – The key of the force field to retrieve.

Returns The retrieved force field source.

Return type ForceFieldSource

retrieve_object(storage_key, expected_type=None)

Retrieves a stored object for the estimators storage system.

Parameters

- **storage_key** (*str*) A unique key that describes where the stored object can be found within the storage system.
- **expected_type** (*type of BaseStoredData, optional*) The expected data type. An exception is raised if the retrieved data doesn't match the type.

Returns

- BaseStoredData, optional The stored object if the object key is found, otherwise None.
- *str*, *optional* The path to the ancillary data if present.

store_field(force_field)

A convenience method for storing ForceFieldSource objects.

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

Returns The unique id of the stored force field.

Return type str

store_object(object_to_store, ancillary_data_path=None)

Store an object in the storage system, returning the key of the stored object. This may be different to *storage_key* depending on whether the same or a similar object was already present in the system.

Parameters

- **object_to_store** (BaseStoredData) The object to store.
- **ancillary_data_path** (*str*, *optional*) The data path to the ancillary directory-like data to store alongside the object if the data type requires one.

Returns The unique key assigned to the stored object.

Return type str

Data Classes

BaseStoredData	A base representation of cached data to be stored by a storage backend.	
HashableStoredData	Represents a class of data objects which can be rapidly compared / indexed by their hash values.	
ForceFieldData	A data container for force field objects which will be saved to disk.	
ReplaceableData	Represents a piece of stored data which can be replaced in a <i>StorageBackend</i> by another piece of data of the same type.	
BaseSimulationData	A base class for classes which will store the outputs of a molecular simulation	
StoredSimulationData	A representation of data which has been cached from a single previous simulation.	
StoredFreeEnergyData	A representation of data which has been cached from an free energy calculation which computed the free energy difference between a start and end state.	

BaseStoredData

class openff.evaluator.storage.data.BaseStoredData

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

The expectation is that stored data may 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 as a file name.
- 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 their file name by the data object.

The ancillary directory-like structure is not required if the data may be suitably stored in the data object itself.

___init__()

Methods

			\sim
7	nı	+	()
		L	()
			~

from_json(file_path)	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>has_ancillary_data()</pre>	Returns whether this data object requires an accom-
	panying data directory-like structure.
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
to_storage_query()	Returns the storage query which would match this
	data object.
validate([attribute_type])	Validate the values of the attributes.

abstract classmethod has_ancillary_data()

Returns whether this data object requires an accompanying data directory-like structure.

Returns True if this class requires an accompanying data directory-like structure.

Return type bool

to_storage_query()

Returns the storage query which would match this data object.

Returns The storage query which would match this data object.

Return type *BaseDataQuery*

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None) Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

```
json(file_path=None, format=False)
```

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

HashableStoredData

class openff.evaluator.storage.data.HashableStoredData

Represents a class of data objects which can be rapidly compared / indexed by their hash values.

__init__()

Methods

___init___()

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>has_ancillary_data()</pre>	Returns whether this data object requires an accom-
	panying data directory-like structure.
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
to_storage_query()	Returns the storage query which would match this
	data object.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

abstract classmethod has_ancillary_data()

Returns whether this data object requires an accompanying data directory-like structure.

Returns True if this class requires an accompanying data directory-like structure.

Return type bool

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

to_storage_query()

Returns the storage query which would match this data object.

Returns The storage query which would match this data object.

Return type BaseDataQuery

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

ForceFieldData

class openff.evaluator.storage.data.ForceFieldData

A data container for force field objects which will be saved to disk.

__init__()

Methods

__init__()

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
= · = ·	
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>has_ancillary_data()</pre>	Returns whether this data object requires an accom-
	panying data directory-like structure.
json([file_path, format])	Creates a JSON representation of this class.
<pre>parse_json(string_contents)</pre>	Parses a typed json string into the corresponding class
	structure.
<pre>to_storage_query()</pre>	
	returns The storage query which would
	match this
validate([attribute_type])	Validate the values of the attributes.

Attributes

force_field_source	The force field source object.

force_field_source

The force field source object. The default value of this attribute is not set and must be set by the user..

Type ForceFieldSource

classmethod has_ancillary_data()

Returns whether this data object requires an accompanying data directory-like structure.

Returns True if this class requires an accompanying data directory-like structure.

Return type bool

to_storage_query()

Returns The storage query which would match this data object.

Return type SimulationDataQuery

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

validate(attribute_type=None)

Validate the values of the attributes. If attribute_type is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

ReplaceableData

class openff.evaluator.storage.data.ReplaceableData

Represents a piece of stored data which can be replaced in a *StorageBackend* by another piece of data of the same type.

This may be the case for example when attempting to store a piece of *StoredSimulationData*, but another piece of data measured from the same calculation and for the same system already exists in the system, but stores less configurations.

__init__()

Methods

__init__()

<pre>from_j son(file_path)</pre>	Create this object from a JSON file.
get_attributes([attribute_type])	Returns all attributes of a specific <i>attribute_type</i> .
has_ancillary_data()	Returns whether this data object requires an accom-
	panying data directory-like structure.
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
<pre>most_information(stored_data_1, stored_data_2)</pre>	Returns the data object with the highest information
	content.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
to_storage_query()	Returns the storage query which would match this
	data object.
	Validate the values of the attributes.
<pre>validate([attribute_type])</pre>	valuate the values of the attributes.

abstract classmethod most_information(stored_data_1, stored_data_2)

Returns the data object with the highest information content.

Parameters

- **stored_data_1** (ReplaceableData) The first piece of data to compare.
- stored_data_2 (ReplaceableData) The second piece of data to compare.

Returns The data object with the highest information content, or *None* if the two pieces of information are incompatible with one another.

Return type ReplaceableData, optional

classmethod from_json(*file_path*)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None) Returns all attributes of a specific attribute_type.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

abstract classmethod has_ancillary_data()

Returns whether this data object requires an accompanying data directory-like structure.

Returns True if this class requires an accompanying data directory-like structure.

Return type bool

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

to_storage_query()

Returns the storage query which would match this data object.

Returns The storage query which would match this data object.

Return type *BaseDataQuery*

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

BaseSimulationData

class openff.evaluator.storage.data.BaseSimulationData

A base class for classes which will store the outputs of a molecular simulation

__init__()

Methods

__init__()

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
has_ancillary_data()	Returns whether this data object requires an accom-
	panying data directory-like structure.
json([file_path, format])	Creates a JSON representation of this class.
<pre>most_information(stored_data_1, stored_data_2)</pre>	Returns the data object with the highest information
	content.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
to_storage_query()	Returns the storage query which would match this
	data object.
validate([attribute_type])	Validate the values of the attributes.

Attributes

force_field_id	The id of the force field parameters used to generate
	the data.
property_phase	The phase of the system (e.g.
source_calculation_id	The server id of the calculation which yielded this
	data.
substance	A description of the composition of the stored sys-
	tem.
thermodynamic_state	The state at which the data was collected.

substance

A description of the composition of the stored system. The default value of this attribute is not set and must be set by the user.

Type Substance

thermodynamic_state

The state at which the data was collected. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

property_phase

The phase of the system (e.g. liquid, gas). The default value of this attribute is not set and must be set by the user.

Type PropertyPhase

source_calculation_id

The server id of the calculation which yielded this data. The default value of this attribute is not set and must be set by the user.

Type str

force_field_id

The id of the force field parameters used to generate the data. The default value of this attribute is not set and must be set by the user.. Type str

classmethod has_ancillary_data()

Returns whether this data object requires an accompanying data directory-like structure.

Returns True if this class requires an accompanying data directory-like structure.

Return type bool

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

abstract classmethod most_information(stored_data_1, stored_data_2)

Returns the data object with the highest information content.

Parameters

- stored_data_1 (ReplaceableData) The first piece of data to compare.
- stored_data_2 (ReplaceableData) The second piece of data to compare.
- **Returns** The data object with the highest information content, or *None* if the two pieces of information are incompatible with one another.

Return type *ReplaceableData*, optional

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

to_storage_query()

Returns the storage query which would match this data object.

Returns The storage query which would match this data object.

Return type BaseDataQuery

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

StoredSimulationData

class openff.evaluator.storage.data.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:

__init__()

Methods

___init__()

from icon(fla noth)	Create this shipst from a ISON fla
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
get_attributes([attribute_type])	Returns all attributes of a specific <i>attribute_type</i> .
has_ancillary_data()	Returns whether this data object requires an accom-
	panying data directory-like structure.
json([file_path, format])	Creates a JSON representation of this class.
<pre>most_information(stored_data_1, stored_data_2)</pre>	Returns the data object with the lowest statisti-
	cal_inefficiency.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
to_storage_query()	
	returns The storage query which would
	match this
	match this
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

coordinate_file_nameThe name of a coordinate file which encodes topology information of the system.force_field_idThe id of the force field parameters used to gene the data.number_of_moleculesThe total number of molecules in the system.observablesA frame of observables collected over the duration the simulation.property_phaseThe phase of the system (e.g.source_calculation_idThe server id of the calculation which yielded	topology info The id of the the data.	formation of the system.
force_field_id The id of the force field parameters used to gene the data. number_of_molecules The total number of molecules in the system. observables A frame of observables collected over the duration the simulation. property_phase The phase of the system (e.g.	The id of the the data.	5
the data. number_of_molecules The total number of molecules in the system. observables A frame of observables collected over the duration. property_phase The phase of the system (e.g.	the data.	e force field parameters used to generate
number_of_molecules The total number of molecules in the system. observables A frame of observables collected over the duration the simulation. property_phase The phase of the system (e.g.		
observables A frame of observables collected over the duration the simulation. property_phase The phase of the system (e.g.	TT1	
property_phase The phase of the system (e.g.	The total nur	mber of molecules in the system.
property_phaseThe phase of the system (e.g.	A frame of o	observables collected over the duration of
	the simulation	on.
source_calculation_id The server id of the calculation which yielded	The phase of	f the system (e.g.
	d The server id	id of the calculation which yielded this
data.	data.	
statistical_inefficiency The statistical inefficiency of the collected data.	ency The statistica	al inefficiency of the collected data.
substance A description of the stored	A description	on of the composition of the stored sys-
tem.	tem.	
thermodynamic_state The state at which the data was collected.	The state at v	which the data was collected.
trajectory_file_name The name of a .dcd trajectory file containing containini	The name of	f a .dcd trajectory file containing config-
urations generated by the simulation.	urations gene	erated by the simulation.

coordinate_file_name

The name of a coordinate file which encodes the topology information of the system. The default value of this attribute is not set and must be set by the user.

Type FilePath

trajectory_file_name

The name of a .dcd trajectory file containing configurations generated by the simulation. The default value of this attribute is not set and must be set by the user.

Type *FilePath*

observables

A frame of observables collected over the duration of the simulation. The default value of this attribute is not set and must be set by the user.

Type ObservableFrame

statistical_inefficiency

The statistical inefficiency of the collected data. The default value of this attribute is not set and must be set by the user.

Type float

number_of_molecules

The total number of molecules in the system. The default value of this attribute is not set and must be set by the user.

Type int

classmethod most_information(stored_data_1, stored_data_2)

Returns the data object with the lowest *statistical_inefficiency*.

Parameters

- stored_data_1 (StoredSimulationData) The first piece of data to compare.
- stored_data_2 (StoredSimulationData) The second piece of data to compare.

Returns

Return type StoredSimulationData

to_storage_query()

Returns The storage query which would match this data object.

Return type *SimulationDataQuery*

force_field_id

The id of the force field parameters used to generate the data. The default value of this attribute is not set and must be set by the user..

Type str

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) - The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None) Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

classmethod has_ancillary_data()

Returns whether this data object requires an accompanying data directory-like structure.

Returns True if this class requires an accompanying data directory-like structure.

Return type bool

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

property_phase

The phase of the system (e.g. liquid, gas). The default value of this attribute is not set and must be set by the user.

Type PropertyPhase

source_calculation_id

The server id of the calculation which yielded this data. The default value of this attribute is not set and must be set by the user.

Type str

substance

A description of the composition of the stored system. The default value of this attribute is not set and must be set by the user.

Type Substance

thermodynamic_state

The state at which the data was collected. The default value of this attribute is not set and must be set by the user.

Type ThermodynamicState

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

StoredFreeEnergyData

class openff.evaluator.storage.data.StoredFreeEnergyData

A representation of data which has been cached from an free energy calculation which computed the free energy difference between a start and end state.

Notes

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

__init__()

Methods

___init__()

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
has_ancillary_data()	Returns whether this data object requires an accom-
	panying data directory-like structure.
json([file_path, format])	Creates a JSON representation of this class.
	continues on next page

	ed from previous page
<pre>most_information(stored_data_1, stored_data_2)</pre>	A comparison function which will always retain both pieces of free energy data.
<pre>parse_json(string_contents)</pre>	Parses a typed json string into the corresponding class structure.
<pre>to_storage_query()</pre>	
	returns The storage query which would
	match this data object.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.
Attailantaa	
Attributes	
end_state_trajectory	The name of a .dcd trajectory file containing config-
	urations generated by the simulation of the end state
	of the system.
force_field_id	The id of the force field parameters used to generate
	the data.
<pre>free_energy_difference</pre>	The free energy difference between the end state and
	the start state.
property_phase	The phase of the system (e.g.
source_calculation_id	The server id of the calculation which yielded this
	data.
start_state_trajectory	The name of a .dcd trajectory file containing config-
	urations generated by the simulation of the start state
	of the system.
substance	A description of the composition of the stored sys-
	tem.
thermodynamic_state	The state at which the data was collected.
topology_file_name	The name of a coordinate file which encodes the
	topology of the system.

Table 206 - continued from previous page

free_energy_difference

The free energy difference between the end state and the start state. The default value of this attribute is not set and must be set by the user.

Type *Observable*

topology_file_name

The name of a coordinate file which encodes the topology of the system. The default value of this attribute is not set and must be set by the user.

Type FilePath

start_state_trajectory

The name of a .dcd trajectory file containing configurations generated by the simulation of the start state of the system. The default value of this attribute is not set and must be set by the user.

Type FilePath

end_state_trajectory

The name of a .dcd trajectory file containing configurations generated by the simulation of the end state of the system. The default value of this attribute is not set and must be set by the user.

Type FilePath

classmethod most_information(*stored_data_1:* openff.evaluator.storage.data.StoredFreeEnergyData, *stored_data_2:* openff.evaluator.storage.data.StoredFreeEnergyData) → Optional[*openff.evaluator.storage.data.StoredFreeEnergyData*]

A comparison function which will always retain both pieces of free energy data. At this time no situation can be envisaged that the same free energy data from exactly the same calculation will be store.

Parameters

- **stored_data_1** The first piece of data to compare.
- **stored_data_2** The second piece of data to compare.

to_storage_query()

Returns The storage query which would match this data object.

Return type FreeEnergyDataQuery

force_field_id

The id of the force field parameters used to generate the data. The default value of this attribute is not set and must be set by the user.

Type str

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

classmethod has_ancillary_data()

Returns whether this data object requires an accompanying data directory-like structure.

Returns True if this class requires an accompanying data directory-like structure.

Return type bool

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

property_phase

The phase of the system (e.g. liquid, gas). The default value of this attribute is not set and must be set by the user..

Type PropertyPhase

source_calculation_id

The server id of the calculation which yielded this data. The default value of this attribute is not set and must be set by the user.

Type str

substance

A description of the composition of the stored system. The default value of this attribute is not set and must be set by the user.

Type Substance

thermodynamic_state

The state at which the data was collected. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

Data Queries

BaseDataQuery	A base class for queries which can be made to a Storage-
	Backend.
SubstanceQuery	A query which focuses on finding data which was col-
	lected for substances with specific traits, e.g which con-
	tains both a solute and solvent, or only a solvent etc.
ForceFieldQuery	A class used to query a StorageBackend for ForceField-
	Data which meet the specified criteria.
BaseSimulationDataQuery	The base class for queries which will retrieve
	BaseSimulationData derived data.
SimulationDataQuery	A class used to query a StorageBackend for
	StoredSimulationData objects which meet the
	specified set of criteria.
FreeEnergyDataQuery	A class used to query a StorageBackend for
	FreeEnergyData objects which meet the specified set
	of criteria.

BaseDataQuery

class openff.evaluator.storage.query.BaseDataQuery

A base class for queries which can be made to a *StorageBackend*.

__init__()

Methods

___init__()

apply(data_object)	Apply this query to a data object.
<pre>data_class()</pre>	The type of data class that this query can be applied
	to.
<pre>from_data_object(data_object)</pre>	Returns the query which would match this data ob-
	ject.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

abstract classmethod data_class()

The type of data class that this query can be applied to.

Returns

Return type type of BaseStoredData

apply(data_object)

Apply this query to a data object.

Parameters data_object (BaseStoredData) - The data object to apply the query to.

Returns The values of the matched parameters of the data object fully matched this query, otherwise *None*.

Return type tuple of Any, optional

classmethod from_data_object(data_object)

Returns the query which would match this data object.

Parameters data_object (BaseStoredData) – The data object to construct the query for.

Returns The query which would match this data object.

Return type cls

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None) Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

SubstanceQuery

class openff.evaluator.storage.query.SubstanceQuery

A query which focuses on finding data which was collected for substances with specific traits, e.g which contains both a solute and solvent, or only a solvent etc.

__init__()

Methods

__init__()

<pre>from_j son(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
validate([attribute_type])	Validate the values of the attributes.

Attributes

components_only	Only match pure data which was collected for one of
	the components in the query substance.

components_only

Only match pure data which was collected for one of the components in the query substance. The default value of this attribute is False.

Type bool

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

ForceFieldQuery

class openff.evaluator.storage.query.ForceFieldQuery

A class used to query a StorageBackend for ForceFieldData which meet the specified criteria.

__init__()

Methods

___init__()

apply(data_object)	Apply this query to a data object.
data_class()	The type of data class that this query can be applied
	to.
<pre>from_data_object(data_object)</pre>	Returns the query which would match this data ob-
	ject.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
validate([attribute_type])	Validate the values of the attributes.

Attributes

classmethod data_class()

The type of data class that this query can be applied to.

Returns

Return type type of BaseStoredData

force_field_source

The force field source to query for. The default value of this attribute is not set. This attribute is optional.

Type ForceFieldSource

apply(data_object)

Apply this query to a data object.

Parameters data_object (BaseStoredData) - The data object to apply the query to.

Returns The values of the matched parameters of the data object fully matched this query, otherwise *None*.

Return type tuple of Any, optional

classmethod from_data_object(data_object)

Returns the query which would match this data object.

Parameters data_object (BaseStoredData) – The data object to construct the query for.

Returns The query which would match this data object.

Return type cls

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

validate(*attribute_type=None*) Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

BaseSimulationDataQuery

class openff.evaluator.storage.query.BaseSimulationDataQuery

The base class for queries which will retrieve BaseSimulationData derived data.

___init__()

Methods

ir	ıi	t	0
 			$- \cdots$

<pre>apply(data_object[, attributes_to_ignore])</pre>	Apply this query to a data object.
data_class()	The type of data class that this query can be applied
	to.
<pre>from_data_object(data_object)</pre>	Returns the query which would match this data ob-
	ject.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

force_field_id	The id of the force field parameters which used to
	generate the data.
property_phase	The phase of the substance (e.g.
source_calculation_id	The server id which should have generated this data.
substance	The substance which the data should have been col-
	lected for.
substance_query	The subset of the <i>substance</i> to query for.
thermodynamic_state	The state at which the data should have been col-
	lected.

substance

The substance which the data should have been collected for. Data for a subset of this substance can be queried for by using the *substance_query* attribute The default value of this attribute is not set. This attribute is *optional*.

Type *Substance*

substance_query

The subset of the *substance* to query for. This option can only be used when the *substance* attribute is set. The default value of this attribute is not set. This attribute is *optional*.

Type SubstanceQuery

thermodynamic_state

The state at which the data should have been collected. The default value of this attribute is not set. This attribute is *optional*.

Type ThermodynamicState

property_phase

The phase of the substance (e.g. liquid, gas). The default value of this attribute is not set. This attribute is *optional*.

Type PropertyPhase

source_calculation_id

The server id which should have generated this data. The default value of this attribute is not set. This attribute is *optional*.

Type str

force_field_id

The id of the force field parameters which used to generate the data. The default value of this attribute is not set. This attribute is *optional*.

Type str

apply(*data_object, attributes_to_ignore=None*) Apply this query to a data object.

Parameters data_object (BaseStoredData) - The data object to apply the query to.

Returns The values of the matched parameters of the data object fully matched this query, otherwise *None*.

Return type tuple of Any, optional

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

abstract classmethod data_class()

The type of data class that this query can be applied to.

Returns

Return type type of BaseStoredData

classmethod from_data_object(data_object)

Returns the query which would match this data object.

Parameters data_object (BaseStoredData) – The data object to construct the query for.

Returns The query which would match this data object.

Return type cls

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

SimulationDataQuery

class openff.evaluator.storage.query.SimulationDataQuery

A class used to query a StorageBackend for StoredSimulationData objects which meet the specified set of criteria.

__init__()

Methods

__init__()

<pre>apply(data_object[, attributes_to_ignore])</pre>	Apply this query to a data object.
data_class()	The type of data class that this query can be applied
	to.
<pre>from_data_object(data_object)</pre>	Returns the query which would match this data ob-
	ject.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
validate([attribute_type])	Validate the values of the attributes.

Attributes

force_field_id	The id of the force field parameters which used to
	generate the data.
number_of_molecules	The total number of molecules in the system.
property_phase	The phase of the substance (e.g.
source_calculation_id	The server id which should have generated this data.
substance	The substance which the data should have been col-
	lected for.
substance_query	The subset of the <i>substance</i> to query for.
	continues on next page

Table 2	17 - continued	from	previous page	
---------	----------------	------	---------------	--

thermodynamic_state	The state at which the data should have been col-
	lected.

classmethod data_class()

The type of data class that this query can be applied to.

Returns

Return type type of BaseStoredData

number_of_molecules

The total number of molecules in the system. The default value of this attribute is not set. This attribute is *optional*.

Type int

apply(*data_object*, *attributes_to_ignore=None*) Apply this query to a data object.

Parameters data_object (BaseStoredData) – The data object to apply the query to.

Returns The values of the matched parameters of the data object fully matched this query, otherwise *None*.

Return type tuple of Any, optional

force_field_id

The id of the force field parameters which used to generate the data. The default value of this attribute is not set. This attribute is *optional*.

Type str

classmethod from_data_object(data_object)

Returns the query which would match this data object.

Parameters data_object (BaseStoredData) – The data object to construct the query for.

Returns The query which would match this data object.

Return type cls

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

property_phase

The phase of the substance (e.g. liquid, gas). The default value of this attribute is not set. This attribute is *optional*.

Type *PropertyPhase*

source_calculation_id

The server id which should have generated this data. The default value of this attribute is not set. This attribute is *optional*.

Type str

substance

The substance which the data should have been collected for. Data for a subset of this substance can be queried for by using the *substance_query* attribute The default value of this attribute is not set. This attribute is *optional*.

Type Substance

substance_query

The subset of the *substance* to query for. This option can only be used when the *substance* attribute is set. The default value of this attribute is not set. This attribute is *optional*.

Type SubstanceQuery

thermodynamic_state

The state at which the data should have been collected. The default value of this attribute is not set. This attribute is *optional*.

Type ThermodynamicState

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

FreeEnergyDataQuery

class openff.evaluator.storage.query.FreeEnergyDataQuery

A class used to query a StorageBackend for FreeEnergyData objects which meet the specified set of criteria.

__init__()

Methods

__init__()

<pre>apply(data_object[, attributes_to_ignore])</pre>	Apply this query to a data object.
<pre>data_class()</pre>	The type of data class that this query can be applied
	to.
<pre>from_data_object(data_object)</pre>	Returns the query which would match this data ob-
	ject.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
validate([attribute_type])	Validate the values of the attributes.

Attributes

force_field_id	The id of the force field parameters which used to
	generate the data.
property_phase	The phase of the substance (e.g.
source_calculation_id	The server id which should have generated this data.
substance	The substance which the data should have been col-
	lected for.
substance_query	The subset of the <i>substance</i> to query for.
thermodynamic_state	The state at which the data should have been col-
	lected.

classmethod data_class()

The type of data class that this query can be applied to.

Returns

Return type type of BaseStoredData

apply(*data_object*, *attributes_to_ignore=None*) Apply this query to a data object.

Parameters data_object (BaseStoredData) - The data object to apply the query to.

Returns The values of the matched parameters of the data object fully matched this query, otherwise *None*.

Return type tuple of Any, optional

force_field_id

The id of the force field parameters which used to generate the data. The default value of this attribute is not set. This attribute is *optional*.

Type str

classmethod from_data_object(data_object)

Returns the query which would match this data object.

Parameters data_object (BaseStoredData) - The data object to construct the query for.

Returns The query which would match this data object.

Return type cls

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) - The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

property_phase

The phase of the substance (e.g. liquid, gas). The default value of this attribute is not set. This attribute is *optional*.

Type *PropertyPhase*

source_calculation_id

The server id which should have generated this data. The default value of this attribute is not set. This attribute is *optional*.

Type str

substance

The substance which the data should have been collected for. Data for a subset of this substance can be queried for by using the *substance_query* attribute The default value of this attribute is not set. This attribute is *optional*.

Type Substance

substance_query

The subset of the *substance* to query for. This option can only be used when the *substance* attribute is set. The default value of this attribute is not set. This attribute is *optional*.

Type SubstanceQuery

thermodynamic_state

The state at which the data should have been collected. The default value of this attribute is not set. This attribute is *optional*.

Type ThermodynamicState

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

Attributes

FilePath	Represents a string file path.
StorageAttribute	A descriptor used to mark attributes of a class as those
	which store information about a cached piece of data.
QueryAttribute	A descriptor used to add additional metadata to at-
	tributes of a storage query.

FilePath

class openff.evaluator.storage.attributes.**FilePath** Represents a string file path.

__init__()

Methods

init()	
capitalize()	Return a capitalized version of the string.
casefold()	Return a version of the string suitable for caseless
	comparisons.
<pre>center(width[, fillchar])</pre>	Return a centered string of length width.
<pre>count(sub[, start[, end]])</pre>	Return the number of non-overlapping occurrences
	of substring sub in string S[start:end].
encode([encoding, errors])	Encode the string using the codec registered for en-
	coding.
	continuos on poyt pago

continues on next page

Table 221 –	continued from previous page
<pre>endswith(suffix[, start[, end]])</pre>	Return True if S ends with the specified suffix, False otherwise.
expandtabs([tabsize])	Return a copy where all tab characters are expanded using spaces.
<pre>find(sub[, start[, end]])</pre>	Return the lowest index in S where substring sub is
	found, such that sub is contained within S[start:end].
<pre>format(*args, **kwargs)</pre>	Return a formatted version of S, using substitutions
	from args and kwargs.
<pre>format_map(mapping)</pre>	Return a formatted version of S, using substitutions
	from mapping.
<pre>index(sub[, start[, end]])</pre>	Return the lowest index in S where substring sub is
	found, such that sub is contained within S[start:end].
isalnum()	Return True if the string is an alpha-numeric string,
	False otherwise.
isalpha()	Return True if the string is an alphabetic string, False
	otherwise.
isascii()	Return True if all characters in the string are ASCII,
	False otherwise.
isdecimal()	Return True if the string is a decimal string, False
	otherwise.
isdigit()	Return True if the string is a digit string, False other-
	wise.
isidentifier()	Return True if the string is a valid Python identifier,
	False otherwise.
islower()	Return True if the string is a lowercase string, False
	otherwise.
isnumeric()	Return True if the string is a numeric string, False
	otherwise.
isprintable()	Return True if the string is printable, False otherwise.
isspace()	Return True if the string is a whitespace string, False
	otherwise.
istitle()	Return True if the string is a title-cased string, False
	otherwise.
isupper()	Return True if the string is an uppercase string, False
	otherwise.
join(iterable, /)	Concatenate any number of strings.
ljust(width[, fillchar])	Return a left-justified string of length width.
lower()	Return a copy of the string converted to lowercase.
lstrip([chars])	Return a copy of the string with leading whitespace
	removed.
<pre>maketrans(x[, y, z])</pre>	Return a translation table usable for str.translate().
<pre>partition(sep, /)</pre>	Partition the string into three parts using the given
	separator.
<pre>replace(old, new[, count])</pre>	Return a copy with all occurrences of substring old
	replaced by new.
<pre>rfind(sub[, start[, end]])</pre>	Return the highest index in S where substring sub is
<pre>rfind(sub[, start[, end]])</pre>	
	found, such that sub is contained within S[start:end].
<pre>rfind(sub[, start[, end]]) rindex(sub[, start[, end]])</pre>	Return the highest index in S where substring sub is

Table 221 – continued from previous page

	continued from previous page
<pre>rpartition(sep, /)</pre>	Partition the string into three parts using the given
	separator.
<pre>rsplit([sep, maxsplit])</pre>	Return a list of the words in the string, using sep as
	the delimiter string.
<pre>rstrip([chars])</pre>	Return a copy of the string with trailing whitespace
	removed.
<pre>split([sep, maxsplit])</pre>	Return a list of the words in the string, using sep as
	the delimiter string.
splitlines([keepends])	Return a list of the lines in the string, breaking at line
	boundaries.
<pre>startswith(prefix[, start[, end]])</pre>	Return True if S starts with the specified prefix, False
	otherwise.
<pre>strip([chars])</pre>	Return a copy of the string with leading and trailing
	whitespace removed.
swapcase()	Convert uppercase characters to lowercase and low-
	ercase characters to uppercase.
title()	Return a version of the string where each word is ti-
	tlecased.
translate(table, /)	Replace each character in the string using the given
	translation table.
upper()	Return a copy of the string converted to uppercase.
zfill(width,/)	Pad a numeric string with zeros on the left, to fill a
	field of the given width.

Table 221 - continued from previous page

capitalize()

Return a capitalized version of the string.

More specifically, make the first character have upper case and the rest lower case.

casefold()

Return a version of the string suitable for caseless comparisons.

center(width, fillchar=' ',/)

Return a centered string of length width.

Padding is done using the specified fill character (default is a space).

count(*sub* |, *start* |, *end* |) \rightarrow int

Return the number of non-overlapping occurrences of substring sub in string S[start:end]. Optional arguments start and end are interpreted as in slice notation.

encode(encoding='utf-8', errors='strict')

Encode the string using the codec registered for encoding.

encoding The encoding in which to encode the string.

errors The error handling scheme to use for encoding errors. The default is 'strict' meaning that encoding errors raise a UnicodeEncodeError. Other possible values are 'ignore', 'replace' and 'xmlcharrefreplace' as well as any other name registered with codecs.register_error that can handle UnicodeEncodeErrors.

endswith(suffix[, start[, end]]) \rightarrow bool

Return True if S ends with the specified suffix, False otherwise. With optional start, test S beginning at that position. With optional end, stop comparing S at that position. suffix can also be a tuple of strings to try.

expandtabs(tabsize=8)

Return a copy where all tab characters are expanded using spaces.

If tabsize is not given, a tab size of 8 characters is assumed.

find(*sub*[, *start*[, *end*]]) \rightarrow int

Return the lowest index in S where substring sub is found, such that sub is contained within S[start:end]. Optional arguments start and end are interpreted as in slice notation.

Return -1 on failure.

format(**args*, ***kwargs*) \rightarrow str

Return a formatted version of S, using substitutions from args and kwargs. The substitutions are identified by braces ('{ and '}').

$format_map(mapping) \rightarrow str$

Return a formatted version of S, using substitutions from mapping. The substitutions are identified by braces ('{ and '}').

index(*sub* $[, start[, end]]) \rightarrow int$

Return the lowest index in S where substring sub is found, such that sub is contained within S[start:end]. Optional arguments start and end are interpreted as in slice notation.

Raises ValueError when the substring is not found.

isalnum()

Return True if the string is an alpha-numeric string, False otherwise.

A string is alpha-numeric if all characters in the string are alpha-numeric and there is at least one character in the string.

isalpha()

Return True if the string is an alphabetic string, False otherwise.

A string is alphabetic if all characters in the string are alphabetic and there is at least one character in the string.

isascii()

Return True if all characters in the string are ASCII, False otherwise.

ASCII characters have code points in the range U+0000-U+007F. Empty string is ASCII too.

isdecimal()

Return True if the string is a decimal string, False otherwise.

A string is a decimal string if all characters in the string are decimal and there is at least one character in the string.

isdigit()

Return True if the string is a digit string, False otherwise.

A string is a digit string if all characters in the string are digits and there is at least one character in the string.

isidentifier()

Return True if the string is a valid Python identifier, False otherwise.

Use keyword.iskeyword() to test for reserved identifiers such as "def" and "class".

islower()

Return True if the string is a lowercase string, False otherwise.

A string is lowercase if all cased characters in the string are lowercase and there is at least one cased character in the string.

isnumeric()

Return True if the string is a numeric string, False otherwise.

A string is numeric if all characters in the string are numeric and there is at least one character in the string.

isprintable()

Return True if the string is printable, False otherwise.

A string is printable if all of its characters are considered printable in repr() or if it is empty.

isspace()

Return True if the string is a whitespace string, False otherwise.

A string is whitespace if all characters in the string are whitespace and there is at least one character in the string.

istitle()

Return True if the string is a title-cased string, False otherwise.

In a title-cased string, upper- and title-case characters may only follow uncased characters and lowercase characters only cased ones.

isupper()

Return True if the string is an uppercase string, False otherwise.

A string is uppercase if all cased characters in the string are uppercase and there is at least one cased character in the string.

join(iterable,/)

Concatenate any number of strings.

The string whose method is called is inserted in between each given string. The result is returned as a new string.

Example: '.'.join(['ab', 'pq', 'rs']) -> 'ab.pq.rs'

ljust(width, fillchar=' ', /)

Return a left-justified string of length width.

Padding is done using the specified fill character (default is a space).

lower()

Return a copy of the string converted to lowercase.

lstrip(chars=None,/)

Return a copy of the string with leading whitespace removed.

If chars is given and not None, remove characters in chars instead.

static maketrans(x, y=None, z=None, /)

Return a translation table usable for str.translate().

If there is only one argument, it must be a dictionary mapping Unicode ordinals (integers) or characters to Unicode ordinals, strings or None. Character keys will be then converted to ordinals. If there are two arguments, they must be strings of equal length, and in the resulting dictionary, each character in x will be mapped to the character at the same position in y. If there is a third argument, it must be a string, whose characters will be mapped to None in the result.

partition(sep,/)

Partition the string into three parts using the given separator.

This will search for the separator in the string. If the separator is found, returns a 3-tuple containing the part before the separator, the separator itself, and the part after it.

If the separator is not found, returns a 3-tuple containing the original string and two empty strings.

```
replace(old, new, count=-1, /)
```

Return a copy with all occurrences of substring old replaced by new.

count Maximum number of occurrences to replace. -1 (the default value) means replace all occurrences.

If the optional argument count is given, only the first count occurrences are replaced.

rfind(*sub*[, *start*[, *end*]]) \rightarrow int

Return the highest index in S where substring sub is found, such that sub is contained within S[start:end]. Optional arguments start and end are interpreted as in slice notation.

Return -1 on failure.

rindex(*sub*[, *start*[, *end*]]) \rightarrow int

Return the highest index in S where substring sub is found, such that sub is contained within S[start:end]. Optional arguments start and end are interpreted as in slice notation.

Raises ValueError when the substring is not found.

rjust(width, fillchar=' ', /)

Return a right-justified string of length width.

Padding is done using the specified fill character (default is a space).

rpartition(*sep*,/)

Partition the string into three parts using the given separator.

This will search for the separator in the string, starting at the end. If the separator is found, returns a 3-tuple containing the part before the separator, the separator itself, and the part after it.

If the separator is not found, returns a 3-tuple containing two empty strings and the original string.

rsplit(*sep=None*, *maxsplit=-1*)

Return a list of the words in the string, using sep as the delimiter string.

sep The delimiter according which to split the string. None (the default value) means split according to any whitespace, and discard empty strings from the result.

maxsplit Maximum number of splits to do. -1 (the default value) means no limit.

Splits are done starting at the end of the string and working to the front.

rstrip(chars=None,/)

Return a copy of the string with trailing whitespace removed.

If chars is given and not None, remove characters in chars instead.

split(sep=None, maxsplit=- 1)

Return a list of the words in the string, using sep as the delimiter string.

sep The delimiter according which to split the string. None (the default value) means split according to any whitespace, and discard empty strings from the result.

maxsplit Maximum number of splits to do. -1 (the default value) means no limit.

splitlines(keepends=False)

Return a list of the lines in the string, breaking at line boundaries.

Line breaks are not included in the resulting list unless keepends is given and true.

startswith(prefix[, start[, end]]) \rightarrow bool

Return True if S starts with the specified prefix, False otherwise. With optional start, test S beginning at that position. With optional end, stop comparing S at that position. prefix can also be a tuple of strings to try.

strip(chars=None,/)

Return a copy of the string with leading and trailing whitespace removed.

If chars is given and not None, remove characters in chars instead.

swapcase()

Convert uppercase characters to lowercase and lowercase characters to uppercase.

title()

Return a version of the string where each word is titlecased.

More specifically, words start with uppercased characters and all remaining cased characters have lower case.

translate(table,/)

Replace each character in the string using the given translation table.

table Translation table, which must be a mapping of Unicode ordinals to Unicode ordinals, strings, or None.

The table must implement lookup/indexing via __getitem__, for instance a dictionary or list. If this operation raises LookupError, the character is left untouched. Characters mapped to None are deleted.

upper()

Return a copy of the string converted to uppercase.

zfill(width,/)

Pad a numeric string with zeros on the left, to fill a field of the given width.

The string is never truncated.

StorageAttribute

class openff.evaluator.storage.attributes.**StorageAttribute**(*docstring*, *type_hint*, *optional=False*) A descriptor used to mark attributes of a class as those which store information about a cached piece of data.

__init__(*docstring*, *type_hint*, *optional=False*) Initializes a new Attribute 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.
- **default_value** (*Any*) The default value for this attribute.
- **optional** (*bool*) Defines whether this is an optional input of a class. If true, the *de*-*fault_value* should be set to *UNDEFINED*.
- read_only (bool) Defines whether this attribute is read-only.

Methods

__init__(docstring, type_hint[, optional]) Initializes a new Attribute object.

QueryAttribute

__init___(docstring, type_hint, optional=False, custom_match=False)
Initializes self.

Parameters custom_match (*bool*) – Whether a custom behaviour will be implemented when matching this attribute against the matching data object attribute.

Methods

__init__(docstring, type_hint[, optional, ...]) Initializes self.

2.32.9 Workflow API

77 1 67	
Workflow	Encapsulates and prepares a workflow which is able to
	estimate a physical property.
WorkflowException	An exception which was raised while executing a work-
	flow protocol.
WorkflowGraph	A hierarchical structure for storing and submitting the
	workflows which will estimate a set of physical proper-
	ties
WorkflowResult	The result of executing a Workflow as part of a Work-
	flowGraph.
Protocol	The base class for a protocol which would form one step
	of a larger property calculation workflow.
ProtocolGraph	A graph of connected protocols which may be executed
	together.
ProtocolGroup	A group of workflow protocols to be executed in one
	batch.
workflow_protocol	A decorator which registers a class as being a protocol
	which may be included in workflows.
register_workflow_protocol	Registers a class as being a protocol which may be in-
	cluded in workflows.

Workflow

class openff.evaluator.workflow.**Workflow**(*global_metadata*, *unique_id=None*) Encapsulates and prepares a workflow which is able to estimate a physical property.

__init__(global_metadata, unique_id=None)
Constructs a new Workflow object.

Parameters

- **global_metadata** (*dict of str and Any*) A dictionary of the metadata which will be made available to each of the workflow protocols through the pseudo "global" scope.
- **unique_id** (*str*, *optional*) A unique identifier to assign to this workflow. This id will be appended to the ids of the protocols of this workflow. If none is provided, one will be chosen at random.

Methods

init(global_metadata[, unique_id])	Constructs a new Workflow object.
<pre>execute([root_directory,])</pre>	Executes the workflow.
<pre>from_schema(schema, metadata[, unique_id])</pre>	Creates a workflow from its schema blueprint, and
	the associated metadata.
<pre>generate_default_metadata(physical_property,</pre>	Generates the default global metadata dictionary.
)	
<i>replace_protocol</i> (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.
to_graph()	Converts this workflow to an executable Workflow-
	Graph.

Attributes

final_value_source	The path to the protocol output which corresponds to
	the estimated value of the property being estimated.
outputs_to_store	A collection of data classes to populate ready to be
	stored by a StorageBackend.
protocols	The protocols in this workflow.
schema	

property protocols

The protocols in this workflow.

Type tuple of Protocol

property final_value_source

The path to the protocol output which corresponds to the estimated value of the property being estimated.

Type ProtocolPath

property outputs_to_store

A collection of data classes to populate ready to be stored by a StorageBackend.

Type dict of str and StorageBackend

replace_protocol(*old_protocol*, *new_protocol*, *update_paths_only=False*)

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** (Protocol or ProtocolPath) The protocol (or its id) to replace.
- new_protocol (Protocol or ProtocolPath) The new protocol (or its id) to use.
- **update_paths_only** (*bool*) Whether only update the *final_value_source*, and *outputs_to_store* attributes, or to also update all of the protocols in *protocols*.

static generate_default_metadata(physical_property, force_field_path,

parameter_gradient_keys=None, target_uncertainty=None) Generates the 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.
- **target_uncertainty** (*openff.evaluator.unit.Quantity*, *optional*) The uncertainty which the property should be estimated to within.

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: openff.evaluator.unit.Quantity The target uncertainty with which properties should be estimated.
- per_component_uncertainty: openff.evaluator.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

to_graph()

Converts this workflow to an executable WorkflowGraph.

Returns The graph representation of this workflow.

Return type WorkflowGraph

classmethod from_schema(schema, metadata, unique_id=None)

Creates a workflow from its schema blueprint, and the associated metadata.

Parameters

- schema (WorkflowSchema) The schema blueprint for this workflow.
- metadata (dict of str and Any) The metadata to make available to the workflow.
- **unique_id** (*str*, *optional*) A unique identifier to assign to this workflow. This id will be appended to the ids of the protocols of this workflow. If none is provided one will be chosen at random.

Returns The created workflow.

Return type cls

execute(*root_directory=''*, *calculation_backend=None*, *compute_resources=None*) Executes the workflow.

Parameters

- **root_directory** (*str*) The directory to execute the graph in.
- **calculation_backend** (CalculationBackend, *optional.*) The backend to execute the graph on. This parameter is mutually exclusive with *compute_resources*.
- **compute_resources** (CalculationBackend, optional.) The compute resources to run using. If None and no *calculation_backend* is specified, the workflow will be executed on a single CPU thread. This parameter is mutually exclusive with *calculation_backend*.
- **Returns** The result of executing this workflow. If executed on a *calculation_backend*, the result will be wrapped in a *Future* object.

Return type WorkflowResult or Future of WorkflowResult

WorkflowException

exception openff.evaluator.workflow.WorkflowException(*message=None*, *protocol_id=None*) An exception which was raised while executing a workflow protocol.

classmethod from_exception(exception)

Initialize this class from an existing exception.

Parameters exception (*Exception*) – The existing exception

Returns The initialized exception object.

Return type cls

classmethod from_json(file_path)
 Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

with_traceback()

Exception.with_traceback(tb) - set self.__traceback__ to tb and return self.

WorkflowGraph

class openff.evaluator.workflow.WorkflowGraph

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

___init__()

Methods

_init__()

<pre>add_workflows(*workflows)</pre>	Insert a set of workflows into the workflow graph.
<pre>execute([root_directory,])</pre>	Executes the workflow graph.

Attributes

protocols	The protocols in this graph.
root_protocols	The ids of the protocols in the group which do not
	take input from the other grouped protocols.

property protocols

The protocols in this graph.

Type dict of str and Protocol

property root_protocols

The ids of the protocols in the group which do not take input from the other grouped protocols.

Type list of str

add_workflows(*workflows)

Insert a set of workflows into the workflow graph.

Parameters workflow (Workflow) - The workflow to insert.

execute(*root_directory=''*, *calculation_backend=None*, *compute_resources=None*) Executes the workflow graph.

Parameters

- root_directory (str) The directory to execute the graph in.
- **calculation_backend** (CalculationBackend, *optional.*) The backend to execute the graph on. This parameter is mutually exclusive with *compute_resources*.
- **compute_resources** (CalculationBackend, optional.) The compute resources to run using. If None and no *calculation_backend* is specified, the workflow will be executed on a single CPU thread. This parameter is mutually exclusive with *calculation_backend*.
- **Returns** The results of executing the graph. If a *calculation_backend* is specified, these results will be wrapped in a *Future*.

Return type list of WorkflowResult or list of Future of WorkflowResult

WorkflowResult

class openff.evaluator.workflow.WorkflowResult

The result of executing a Workflow as part of a WorkflowGraph.

__init__()

Methods

_init__()

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
validate([attribute_type])	Validate the values of the attributes.

Attributes

Paths to the data objects to store.
Any exceptions raised by the layer while estimating
the property.
The gradients of the estimated value with respect to
the specified force field parameters.
The estimated value of the property and the uncer-
tainty in that value.
The id of the workflow associated with this result.

workflow_id

The id of the workflow associated with this result. The default value of this attribute is not set and must be set by the user.

Type str

value

The estimated value of the property and the uncertainty in that value. The default value of this attribute is not set. This attribute is *optional*.

Type Measurement

gradients

The gradients of the estimated value with respect to the specified force field parameters. The default value of this attribute is [].

Type list

exceptions

Any exceptions raised by the layer while estimating the property. The default value of this attribute is [].

Type list

data_to_store

Paths to the data objects to store. The default value of this attribute is [].

Type list

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

Protocol

```
class openff.evaluator.workflow.Protocol(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 openff.evaluator.protocols.openmm import OpenMMSimulation
>>>
>>> npt_equilibration = OpenMMSimulation('npt_equilibration')
>>> npt_equilibration.ensemble = OpenMMSimulation.Ensemble.NPT
```

or from the output of another protocol, pointed to by a ProtocolPath

```
>>> npt_production = OpenMMSimulation('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.

__init__(protocol_id)

Methods

___init___(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).

continues on next page

	lice nom previous page
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Table 231 - continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.

allow_merging

Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is **True**.

Type bool

property schema

A serializable schema for this object.

Type ProtocolSchema

property required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

can_merge(other, path_replacements=None)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*Protocol*) 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 Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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 *in-put_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 (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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

ProtocolGraph

class openff.evaluator.workflow.ProtocolGraph

A graph of connected protocols which may be executed together.

__init__()

Methods

___init__()

<pre>add_protocols(*protocols[,])</pre>	Adds a set of protocols to the graph.
<pre>execute([root_directory,])</pre>	Execute the protocol graph in the specified directory,
	and either using a CalculationBackend, or using a
	specified set of compute resources.

Attributes

protocols	The protocols in this graph.
root_protocols	The ids of the protocols in the group which do not
	take input from the other grouped protocols.

property protocols

The protocols in this graph.

Type dict of str and Protocol

property root_protocols

The ids of the protocols in the group which do not take input from the other grouped protocols.

Type list of str

$\verb+add_protocols(*protocols, allow_external_dependencies=False)$

Adds a set of protocols to the graph.

Parameters

- protocols (tuple of Protocol) The protocols to add.
- **allow_external_dependencies** (*bool*) If *False*, an exception will be raised if a protocol has a dependency outside of this graph.
- **Returns** A mapping between the original protocols and protocols which were merged over the course of adding the new protocols.

Return type dict of str and str

execute(root_directory=", calculation_backend=None, compute_resources=None,

enable_checkpointing=True, safe_exceptions=True)

Execute the protocol graph in the specified directory, and either using a *CalculationBackend*, or using a specified set of compute resources.

Parameters

- root_directory (str) The directory to execute the graph in.
- **calculation_backend** (CalculationBackend, optional.) The backend to execute the graph on. This parameter is mutually exclusive with *compute_resources*.
- **compute_resources** (CalculationBackend, *optional.*) The compute resources to run using. This parameter is mutually exclusive with *calculation_backend*.
- **enable_checkpointing** (*bool*) If enabled, protocols will not be executed more than once if the output from their previous execution is found.
- **safe_exceptions** (*bool*) If true, exceptions will be serialized into the results file rather than directly raised, otherwise, the exception will be raised as normal.
- **Returns** The paths to the JSON serialized outputs of the executed protocols. If executed using a calculation backend, these will be *Future* objects which will return the output paths on calling *future.result()*.

Return type dict of str and str or Future

ProtocolGroup

class openff.evaluator.workflow.ProtocolGroup(protocol_id)

A group of workflow 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(protocol_id)	Constructs a new ProtocolGroup.
<pre>add_protocols(*protocols)</pre>	Add protocols to this group.
<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
	continues on next page

	aca nom providad page
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Store the uuid of the calculation this protocol belongs
	to
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Table 235 - continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
protocols	A dictionary of the protocols in this groups, where
	the dictionary key is the protocol id, and the value is
	the protocol itself.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.

property required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

property protocols

A dictionary of the protocols in this groups, where the dictionary key is the protocol id, and the value is the protocol itself.

Notes

This property should not be altered. Use add_protocols to add new protocols to the group.

Type dict of str and Protocol

```
add_protocols(*protocols)
```

Add protocols to this group.

Parameters protocols (Protocol) – The protocols 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.

can_merge(other, path_replacements=None)

Determines whether this protocol can be merged with another.

Parameters

- **other** (*Protocol*) 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 Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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 *in-put_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 (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

allow_merging

Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is **True**.

Type bool

execute(directory=", available_resources=None)
Execute the protocol.

Parameters

- directory (str) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

property schema

A serializable schema for this object.

Type ProtocolSchema

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.
- validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

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** (*boo1*) – 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

workflow_protocol

openff.evaluator.workflow.workflow_protocol()

A decorator which registers a class as being a protocol which may be included in workflows.

register_workflow_protocol

openff.evaluator.workflow.register_workflow_protocol(protocol_class)

Registers a class as being a protocol which may be included in workflows.

Schemas

ProtocolSchema	A json serializable representation of a workflow proto-
	col.
ProtocolGroupSchema	A json serializable representation of a workflow protocol
	group.
ProtocolReplicator	A protocol replicator contains the information necessary
	to replicate parts of a property estimation workflow.
WorkflowSchema	The schematic for a property estimation workflow.

ProtocolSchema

A json serializable representation of a workflow protocol.

__init__(unique_id=None, protocol_type=None, inputs=None)

Methods

__init__([unique_id, protocol_type, inputs])

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
to_protocol()	Creates a new protocol object from this schema.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

id	The unique id associated with the protocol.
inputs	The inputs to the protocol.
type	The type of protocol associated with this schema.

id

The unique id associated with the protocol. The default value of this attribute is not set and must be set by the user.

Type str

type

The type of protocol associated with this schema. The default value of this attribute is not set and must be set by the user.. This attribute is *read-only*.

Type str

inputs

The inputs to the protocol. The default value of this attribute is not set and must be set by the user. This attribute is *read-only*.

Type dict

to_protocol()

Creates a new protocol object from this schema.

Returns The protocol created from this schema.

Return type Protocol

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

ProtocolGroupSchema

class openff.evaluator.workflow.schemas.**ProtocolGroupSchema**(*unique_id=None*,

protocol_type=None, inputs=None,
protocol_schemas=None)

A json serializable representation of a workflow protocol group.

__init__(unique_id=None, protocol_type=None, inputs=None, protocol_schemas=None)

Methods

__init__([unique_id, protocol_type, inputs, ...])

<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
to_protocol()	Creates a new protocol object from this schema.
validate([attribute_type])	Validate the values of the attributes.

Attributes

id	The unique id associated with the protocol.
inputs	The inputs to the protocol.
protocol_schemas	The schemas of the protocols within this group.
type	The type of protocol associated with this schema.

protocol_schemas

The schemas of the protocols within this group. The default value of this attribute is not set and must be set by the user.. This attribute is *read-only*.

Type dict

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

id

The unique id associated with the protocol. The default value of this attribute is not set and must be set by the user.

Type str

inputs

The inputs to the protocol. The default value of this attribute is not set and must be set by the user. This attribute is *read-only*.

Type dict

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

to_protocol()

Creates a new protocol object from this schema.

Returns The protocol created from this schema.

Return type Protocol

type

The type of protocol associated with this schema. The default value of this attribute is not set and must be set by the user.. This attribute is *read-only*.

Type str

ProtocolReplicator

class openff.evaluator.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

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

Attributes

placeholder_id	The string which protocols to be replicated should in-
	clude 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 Protocol) 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 Protocol 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 Protocol*) 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.

classmethod from_json(*file_path*) Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

WorkflowSchema

class openff.evaluator.workflow.schemas.WorkflowSchema

The schematic for a property estimation workflow.

___init__()

Methods

__init__()

Create this object from a JSON file.
Returns all attributes of a specific <i>attribute_type</i> .
Creates a JSON representation of this class.
Parses a typed json string into the corresponding class
structure.
Replaces protocols with given types with other pro-
tocols of specified replacements.
Validate the values of the attributes.

Attributes

final_value_source	A reference to which protocol output corresponds to
	the estimated value of the property.
outputs_to_store	A collection of data classes to populate ready to be
	stored by a StorageBackend.
protocol_replicators	A set of replicators which will replicate parts of the
	workflow.
protocol_schemas	The schemas for the protocols which will make up the
	workflow.

protocol_schemas

The schemas for the protocols which will make up the workflow. The default value of this attribute is [].

Type list

protocol_replicators

A set of replicators which will replicate parts of the workflow. The default value of this attribute is not set. This attribute is *optional*.

Type list

final_value_source

A reference to which protocol output corresponds to the estimated value of the property. The default value of this attribute is not set. This attribute is *optional*.

Type ProtocolPath

outputs_to_store

A collection of data classes to populate ready to be stored by a *StorageBackend*. The default value of this attribute is not set. This attribute is *optional*.

Type dict

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, optional*) 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(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

Attributes

BaseMergeBehaviour	A base class for enums which will describes how at-
	tributes should be handled when attempting to merge
	similar protocols.
MergeBehaviour	A enum which describes how attributes should be han-
	dled when attempting to merge similar protocols.
InequalityMergeBehaviour	A enum which describes how attributes which can be
	compared with inequalities should be merged.
InputAttribute	A descriptor used to mark an attribute of an object as an
	input to that object.
OutputAttribute	A descriptor used to mark an attribute of an as an output
	of that object.

BaseMergeBehaviour

```
class openff.evaluator.workflow.attributes.BaseMergeBehaviour(value)
```

A base class for enums which will describes how attributes should be handled when attempting to merge similar protocols.

__init__()

MergeBehaviour

class openff.evaluator.workflow.attributes.MergeBehaviour(value)

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.
- Custom: This attribute will be ignored by the built-in merging code such that user specified behavior can be implemented.

__init__()

Attributes

ExactlyEqual

Custom

InequalityMergeBehaviour

class openff.evaluator.workflow.attributes.InequalityMergeBehaviour(value)

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__()

Attributes

SmallestValue

LargestValue

InputAttribute

class openff.evaluator.workflow.attributes.InputAttribute(docstring, type_hint, default_value,

optional=False,

merge_behavior=MergeBehaviour.ExactlyEqual)

A descriptor used to mark an attribute of an object as an input to that object.

An attribute can either be set with a value directly, or it can also be set to a *ProtocolPath* to be set be the workflow manager.

Examples

To mark an attribute as an input:

```
>>> from openff.evaluator.attributes import AttributeClass
>>> from openff.evaluator.workflow.attributes import InputAttribute
>>>
class MyObject(AttributeClass):
>>>
my_input = InputAttribute(
>>> docstring='An input will be used.',
>>> type_hint=float,
>>> default_value=0.1
>>> )
```

__init__(docstring, type_hint, default_value, optional=False, merge_behavior=MergeBehaviour.ExactlyEqual) Initializes a new InputAttribute object.

Parameters merge_behavior (BaseMergeBehaviour) – An enum describing how this input should be handled when considering whether to, and actually merging two different objects.

Methods

__*init__*(docstring, type_hint, default_value) Initializes a new InputAttribute object.

OutputAttribute

class openff.evaluator.workflow.attributes.OutputAttribute(docstring, type_hint)

A descriptor used to mark an attribute of an as an output of that object. This attribute is expected to be populated by the object itself, rather than be set externally.

Examples

To mark an attribute as an output:

```
>>> from openff.evaluator.attributes import AttributeClass
>>> from openff.evaluator.workflow.attributes import OutputAttribute
>>>
class MyObject(AttributeClass):
>>>
my_output = OutputAttribute(
>>> docstring='An output that will be filled.',
>>> type_hint=float
>>> )
```

__init__(*docstring*, *type_hint*) Initializes a new OutputAttribute object.

Methods

outAttribute object.
ļ

Placeholder Values

ReplicatorValue	A placeholder value which will be set by a protocol repli- cator with the specified id.
ProtocolPath	Represents a pointer to the output of another protocol.

ReplicatorValue

class openff.evaluator.workflow.utils.**ReplicatorValue**(*replicator_id=''*) A placeholder value which will be set by a protocol replicator with the specified id.

A placeholder value which will be set by a protocol replicator with

__init__(replicator_id=")

Constructs a new ReplicatorValue object

Parameters replicator_id (*str*) – The id of the replicator which will set this value.

Methods

init([replicator_id])	Constructs a new ReplicatorValue object

ProtocolPath

- **class** openff.evaluator.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

init([property_name])	Constructs a new ProtocolPath object.
append_uuid(uuid)	Appends a uuid to each of the protocol id's in the path
copy()	Returns a copy of this path.
from_string(existing_path_string)	
<pre>pop_next_in_path()</pre>	Pops and then returns the leading protocol id from the
	path.
<pre>prepend_protocol_id(id_to_prepend)</pre>	Prepend a new protocol id onto the front of the path.
<pre>replace_protocol(old_id, new_id)</pre>	Redirect the input to point at a new protocol.

Attributes

full_path	The full path referenced by this object.
is_global	
last_protocol	The end protocol id of the path.
path_separator	
property_name	The property name pointed to by the path.
property_separator	
protocol_ids	The ids of the protocols referenced by this object.
protocol_path	The full path referenced by this object excluding the
	property name.
start_protocol	The leading protocol id of the path.

property property_name

The property name pointed to by the path.

Type str

property protocol_ids

The ids of the protocols referenced by this object.

Type tuple of str

property start_protocol

The leading protocol id of the path.

Type str

property last_protocol

The end 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

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.

copy()

Returns a copy of this path.

2.32.10 Built-in Workflow Protocols

Analysis

BaseAverageObservable	An abstract base class for protocols which will calculate
	the average value of an observable and its uncertainty
	via bootstrapping.
AverageObservable	Computes the average value of an observable as well as
	bootstrapped uncertainties for the average.
	continues on next page

continues on next page

Table 255 – continued from previous page	
AverageDielectricConstant	Computes the average value of the dielectric constant from a set of dipole moments (M) and volumes (V) sampled over the course of a molecular simulation such that eps = $1 + ([1].$
AverageFreeEnergies	A protocol which computes the Boltzmann weighted average ($G^{\circ} = -RT \times Log[_{n} \exp(-G^{\circ}_{n})]$) of a set of free energies which were measured at the same thermodynamic state.
ComputeDipoleMoments	A protocol which will compute the dipole moment for each configuration in a trajectory and for a given param- eterized system.
BaseDecorrelateProtocol	An abstract base class for protocols which will subsample a set of data, yielding only equilibrated, uncorrelated data.
DecorrelateTrajectory	A protocol which will subsample frames from a trajec- tory, yielding only uncorrelated frames as determined from a provided statistical inefficiency and equilibration time.
DecorrelateObservables	A protocol which will subsample a trajectory of observ- ables, yielding only uncorrelated entries as determined from a provided statistical inefficiency and equilibration time.

al f., ...

BaseAverageObservable

class openff.evaluator.protocols.analysis.BaseAverageObservable(protocol_id)

An abstract base class for protocols which will calculate the average value of an observable and its uncertainty via bootstrapping.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
<pre>get_value(reference_path)</pre>	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
	continues on next page

continues on next page

	1 1 5
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Table 256 – continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
bootstrap_iterations	Input - The number of bootstrap iterations to per-
	form.
<pre>bootstrap_sample_size</pre>	Input - The relative sample size to use for bootstrap-
	ping.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
potential_energies	Input - The potential energies which were evaluated
	at the same configurations and using the same force
	field parameters as the observable to average.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
thermodynamic_state	Input - The state at which the observables were com-
	puted.
time_series_statistics	Output - Statistics about the observables from which
	the average was computed. These include the statisti-
	cal inefficiency and the index after which the observ-
	ables have become stationary (i.e.
value	Output - The average value of the observable.

bootstrap_iterations

Input - The number of bootstrap iterations to perform. The default value of this attribute is 250.

Type int

bootstrap_sample_size

Input - The relative sample size to use for bootstrapping. The default value of this attribute is 1.0.

Type float

thermodynamic_state

Input - The state at which the observables were computed. This is required to compute ensemble averages of the gradients of the observable with respect to force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type ThermodynamicState

potential_energies

Input - The potential energies which were evaluated at the same configurations and using the same force

field parameters as the observable to average. This is required to compute ensemble averages of the gradients of the observable with respect to force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type ObservableArray

value

Output - The average value of the observable. The default value of this attribute is not set and must be set by the user..

Type Observable

time_series_statistics

Output - Statistics about the observables from which the average was computed. These include the statistical inefficiency and the index after which the observables have become stationary (i.e. equilibrated). The default value of this attribute is not set and must be set by the user.

Type TimeSeriesStatistics

allow_merging

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 providied.

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

```
json(file_path=None, format=False)
```

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

AverageObservable

class openff.evaluator.protocols.analysis.**AverageObservable**(*protocol_id*) Computes the average value of an observable as well as bootstrapped uncertainties for the average.

__init__(protocol_id)

Methods

__init__(protocol_id)

Applies a <i>ProtocolReplicator</i> to this protocol.
Determines whether this protocol can be merged with
another.
Execute the protocol.
Create this object from a JSON file.
Initializes a protocol from it's schema definition.
Returns all attributes of a specific <i>attribute_type</i> .
Returns one of this protocols, or any of its children's,
attributes directly (rather than its value).
Returns the value of one of this protocols inputs / out-
puts.
Returns a dictionary of references to the protocols
which one of this protocols inputs (specified by in-
<i>put_path</i>) takes its value from.
Creates a JSON representation of this class.
Merges another Protocol with this one.
Parses a typed json string into the corresponding class
structure.
Finds each input which came from a given protocol
Prepend a unique identifier to this protocols id.
Sets the value of one of this protocols inputs.
Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
bootstrap_iterations	Input - The number of bootstrap iterations to per-
	form.
<pre>bootstrap_sample_size</pre>	Input - The relative sample size to use for bootstrap-
	ping.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
divisor	Input - A value to divide the statistic by.
id	The unique id of this protocol.
observable	Input - The file path to the observable which should
	be averaged.
outputs	A dictionary of the outputs of this property.
potential_energies	Input - The potential energies which were evaluated
	at the same configurations and using the same force
	field parameters as the observable to average.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
thermodynamic_state	Input - The state at which the observables were com-
	puted.

continues on next page

Table 200 – continued nom previous page	
time_series_statistics	Output - Statistics about the observables from which
	the average was computed. These include the statisti-
	cal inefficiency and the index after which the observ-
	ables have become stationary (i.e.
value	Output - The average value of the observable.

Table 259 - continued from previous page

observable

Input - The file path to the observable which should be averaged. The default value of this attribute is not set and must be set by the user.

Type *ObservableArray*

divisor

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, openff.evaluator.utils.units.Quantity]

allow_merging

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

Input - The number of bootstrap iterations to perform. The default value of this attribute is 250.

Type int

bootstrap_sample_size

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(*attribute_type=None*) Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user...

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

potential_energies

Input - The potential energies which were evaluated at the same configurations and using the same force field parameters as the observable to average. This is required to compute ensemble averages of the gradients of the observable with respect to force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type *ObservableArray*

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

thermodynamic_state

Input - The state at which the observables were computed. This is required to compute ensemble averages of the gradients of the observable with respect to force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type *ThermodynamicState*

time_series_statistics

Output - Statistics about the observables from which the average was computed. These include the statistical inefficiency and the index after which the observables have become stationary (i.e. equilibrated). The default value of this attribute is not set and must be set by the user.

Type TimeSeriesStatistics

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

value

Output - The average value of the observable. The default value of this attribute is not set and must be set by the user.

Type Observable

AverageDielectricConstant

References

[1] A. Glattli, X. Daura and W. F. van Gunsteren. Derivation of an improved simple point charge model for liquid water: SPC/A and SPC/L. J. Chem. Phys. 116(22): 9811-9828, 2002

__init__(protocol_id)

Methods

___init___(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
	continues on port page

continues on next page

	continuou nom providuo pago
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
<pre>get_value(reference_path)</pre>	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Table 260 - continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
bootstrap_iterations	Input - The number of bootstrap iterations to per-
	form.
<pre>bootstrap_sample_size</pre>	Input - The relative sample size to use for bootstrap-
	ping.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
dipole_moments	Input - The dipole moments of each sampled config-
	uration.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
potential_energies	Input - The potential energies which were evaluated
	at the same configurations and using the same force
	field parameters as the observable to average.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
thermodynamic_state	Input - The state at which the observables were com-
	puted.
time_series_statistics	Output - Statistics about the observables from which
	the average was computed. These include the statisti-
	cal inefficiency and the index after which the observ-
	ables have become stationary (i.e.
value	Output - The average value of the observable.
volumes	Input - The volume of each sampled configuration.

dipole_moments

Input - The dipole moments of each sampled configuration. The default value of this attribute is not set and must be set by the user.

Type ObservableArray

volumes

Input - The volume of each sampled configuration. The default value of this attribute is not set and must be set by the user.

Type *ObservableArray*

allow_merging

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 providied.

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

Input - The number of bootstrap iterations to perform. The default value of this attribute is 250.

Type int

bootstrap_sample_size

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

```
json(file_path=None, format=False)
```

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

potential_energies

Input - The potential energies which were evaluated at the same configurations and using the same force field parameters as the observable to average. This is required to compute ensemble averages of the gradients of the observable with respect to force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type ObservableArray

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

thermodynamic_state

Input - The state at which the observables were computed. This is required to compute ensemble averages of the gradients of the observable with respect to force field parameters. The default value of this attribute is not set. This attribute is *optional*.

Type ThermodynamicState

time_series_statistics

Output - Statistics about the observables from which the average was computed. These include the statistical inefficiency and the index after which the observables have become stationary (i.e. equilibrated). The default value of this attribute is not set and must be set by the user.

Type TimeSeriesStatistics

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

value

Output - The average value of the observable. The default value of this attribute is not set and must be set by the user.

Type Observable

AverageFreeEnergies

class openff.evaluator.protocols.analysis.AverageFreeEnergies(protocol_id)

A protocol which computes the Boltzmann weighted average ($G^\circ = -RT \times Log[_{n} \exp(-G^\circ_{n})]$) of a set of free energies which were measured at the same thermodynamic state. Confidence intervals are computed by bootstrapping with replacement.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
from_json(file_path)	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
bootstrap_cycles	Input - The number of bootstrap cycles to perform
	when estimating the uncertainty in the combined free
	energies.
confidence_intervals	Output - The 95% confidence intervals on the aver-
	age free energy.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
result	Output - The sum of the values.
schema	A serializable schema for this object.
thermodynamic_state	Input - The thermodynamic state at which the free
	energies were measured.
values	Input - The values to add together.

values: List[openff.evaluator.utils.observables.Observable]

Input - The values to add together. The default value of this attribute is not set and must be set by the user.

Type list

thermodynamic_state

Input - The thermodynamic state at which the free energies were measured. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

bootstrap_cycles

Input - The number of bootstrap cycles to perform when estimating the uncertainty in the combined free energies. The default value of this attribute is **2000**.

Type int

result

Output - The sum of the values. The default value of this attribute is not set and must be set by the user.

Type Observable

confidence_intervals

Output - The 95% confidence intervals on the average free energy. The default value of this attribute is not set and must be set by the user.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

allow_merging

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 providied.

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 (Protocol) 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

execute(directory=", available_resources=None)
Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

ComputeDipoleMoments

class openff.evaluator.protocols.analysis.**ComputeDipoleMoments**(*protocol_id*) A protocol which will compute the dipole moment for each configuration in a trajectory and for a given param-

eterized system.

__init__(protocol_id)

Methods

__init__(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).

continues on next page

get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Table 264 – continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
dipole_moments	Output - The computed dipole moments.
gradient_parameters	Input - An optional list of parameters to differentiate
	the dipole moments with respect to.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
parameterized_system	Input - The parameterized system which encodes the
	charge on each atom in the system.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
trajectory_path	Input - The file path to the trajectory of configura-
	tions.

parameterized_system

Input - The parameterized system which encodes the charge on each atom in the system. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

trajectory_path

Input - The file path to the trajectory of configurations. The default value of this attribute is not set and must be set by the user..

Type str

gradient_parameters

Input - An optional list of parameters to differentiate the dipole moments with respect to.

Type list

dipole_moments

Output - The computed dipole moments. The default value of this attribute is not set and must be set by the user.

Type ObservableArray

allow_merging

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(*file_path*)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

BaseDecorrelateProtocol

class openff.evaluator.protocols.analysis.BaseDecorrelateProtocol(protocol_id)

An abstract base class for protocols which will subsample a set of data, yielding only equilibrated, uncorrelated data.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
<pre>get_value(reference_path)</pre>	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
time_series_statistics	Input - Statistics about the data to decorrelate.

time_series_statistics:

Union[openff.evaluator.utils.timeseries.TimeSeriesStatistics,

List[openff.evaluator.utils.timeseries.TimeSeriesStatistics]]

Input - Statistics about the data to decorrelate. This should include the statistical inefficiency and the index after which the observables have become stationary (i.e. equilibrated). If a list of such statistics are provided it will be assumed that multiple time series which have been joined together are being decorrelated and hence will each be decorrelated separately. The default value of this attribute is not set and must be set by the user..

Type typing.Union[list, openff.evaluator.utils.timeseries.TimeSeriesStatistics]

allow_merging

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 providied.

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** (Protocol) 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

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(*file_path*) Create this object from a JSON file.

reate this object from a JSON me.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

DecorrelateTrajectory

class openff.evaluator.protocols.analysis.DecorrelateTrajectory(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)

Methods

___init___(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
<pre>input_coordinate_file</pre>	Input - The file path to the starting coordinates of a
	trajectory.
input_trajectory_path	Input - The file path to the trajectory to subsample.
output_trajectory_path	Output - The file path to the subsampled trajectory.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
time_series_statistics	Input - Statistics about the data to decorrelate.

input_coordinate_file

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

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

Output - The file path to the subsampled trajectory. The default value of this attribute is not set and must be set by the user.

Type str

allow_merging

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 providied.

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** (Protocol) – 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

time_series_statistics:

Union[openff.evaluator.utils.timeseries.TimeSeriesStatistics,

List[openff.evaluator.utils.timeseries.TimeSeriesStatistics]]

Input - Statistics about the data to decorrelate. This should include the statistical inefficiency and the index after which the observables have become stationary (i.e. equilibrated). If a list of such statistics are provided it will be assumed that multiple time series which have been joined together are being decorrelated and hence will each be decorrelated separately. The default value of this attribute is not set and must be set by the user..

Type typing.Union[list, openff.evaluator.utils.timeseries.TimeSeriesStatistics]

validate(attribute_type=None)

Validate the values of the attributes. If attribute_type is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

DecorrelateObservables

class openff.evaluator.protocols.analysis.DecorrelateObservables(protocol_id)

A protocol which will subsample a trajectory of observables, yielding only uncorrelated entries as determined from a provided statistical inefficiency and equilibration time.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
set_value(reference_path, value)	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
input_observables	Input - The observables to decorrelate.
output_observables	Output - The decorrelated observables.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
time_series_statistics	Input - Statistics about the data to decorrelate.

input_observables

Input - The observables to decorrelate. The default value of this attribute is not set and must be set by the user.

Type typing.Union[openff.evaluator.utils.observables.ObservableArray, openff.evaluator.utils.observables.ObservableFrame]

output_observables

Output - The decorrelated observables. The default value of this attribute is not set and must be set by the user.

Type typing.Union[*openff.evaluator.utils.observables.ObservableArray*, *openff.evaluator.utils.observables.ObservableFrame*]

allow_merging

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 providied.

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** (Protocol) 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

execute(directory=", available_resources=None)
Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

time_series_statistics:

Union[openff.evaluator.utils.timeseries.TimeSeriesStatistics,

List[openff.evaluator.utils.timeseries.TimeSeriesStatistics]]

Input - Statistics about the data to decorrelate. This should include the statistical inefficiency and the index after which the observables have become stationary (i.e. equilibrated). If a list of such statistics are provided it will be assumed that multiple time series which have been joined together are being decorrelated and hence will each be decorrelated separately. The default value of this attribute is not set and must be set by the user.

Type typing.Union[list, openff.evaluator.utils.timeseries.TimeSeriesStatistics]

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

Coordinate Generation

BuildCoordinatesPackmol	Creates a set of 3D coordinates with a specified compo- sition using the PACKMOL package.
SolvateExistingStructure	Solvates a set of 3D coordinates with a specified solvent using the PACKMOL package.
BuildDockedCoordinates	Creates a set of coordinates for a ligand bound to some receptor.

BuildCoordinatesPackmol

__init__(protocol_id)

Methods

__init__(protocol_id)

	$\frac{1}{1} + \frac{1}{1} + \frac{1}$
<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
get_class_attribute(reference_path)	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
assigned_residue_names	Output - The residue names which were assigned to
	each of the components.
<pre>box_aspect_ratio</pre>	Input - The aspect ratio of the simulation box.
coordinate_file_path	Output - The file path to the created PDB coordinate
	file.
count_exact_amount	Input - Whether components present in an exact
	amount (i.e.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
mass_density	Input - The target density of the created system.
	continues on next page

class openff.evaluator.protocols.coordinates.**BuildCoordinatesPackmol**(*protocol_id*) Creates a set of 3D coordinates with a specified composition using the PACKMOL package.

Table 274 – continued from previous page	
max_molecules	Input - The maximum number of molecules to be
	added to the system.
<pre>output_number_of_molecules</pre>	Output - The number of molecules in the created sys-
	tem.
output_substance	Output - The substance which was built by packmol.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
<pre>retain_packmol_files</pre>	Input - If True, packmol will not delete all of the tem-
	porary files it creates while building the coordinates.
schema	A serializable schema for this object.
substance	Input - The composition of the system to build.
tolerance	Input - The packmol distance tolerance in units com-
	patible with angstroms.
verbose_packmol	Input - If True, packmol will print verbose informa-
	tion to the logger The default value of this attribute is
	False.

Table 274 – continued from previous page

max_molecules

Input - The maximum number of molecules to be added to the system. The default value of this attribute is 1000.

Type int

count_exact_amount

Input - Whether components present in an exact amount (i.e. defined with an ExactAmount) should be considered when apply the maximum number of molecules constraint. This may be set false, for example, when building a separate solvated protein (n = 1) and solvated protein + ligand complex (n = 2) system but wish for both systems to have the same number of solvent molecules. The default value of this attribute is True.

Type bool

mass_density

Input - The target density of the created system. The default value of this attribute is 0.95 g / ml.

Type Quantity

box_aspect_ratio

Input - The aspect ratio of the simulation box. The default value of this attribute is [1.0, 1.0, 1.0].

Type list

substance

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

tolerance

Input - The packmol distance tolerance in units compatible with angstroms. The default value of this attribute is 2.0 Å.

Type Quantity

verbose_packmol

Input - If True, packmol will print verbose information to the logger The default value of this attribute is False.

Type bool

retain_packmol_files

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

Output - The number of molecules in the created system. This may be less than maximum requested due to rounding of mole fractions The default value of this attribute is not set and must be set by the user.

Type int

output_substance

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. The default value of this attribute is not set and must be set by the user.

Type Substance

assigned_residue_names

Output - The residue names which were assigned to each of the components. Each key corresponds to a component identifier. The default value of this attribute is not set and must be set by the user.

Type dict

coordinate_file_path

Output - The file path to the created PDB coordinate file. The default value of this attribute is not set and must be set by the user..

Type str

allow_merging

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 providied.

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

SolvateExistingStructure

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
assigned_residue_names	Output - The residue names which were assigned to
	each of the components.
box_aspect_ratio	Input - The aspect ratio of the simulation box.
center_solute_in_box	Input - If <i>True</i> , the solute to solvate will be centered
	in the simulation box.
coordinate_file_path	Output - The file path to the created PDB coordinate
	file.
count_exact_amount	Input - Whether components present in an exact
	amount (i.e.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
	continuos on noxt nago

continues on next page

class openff.evaluator.protocols.coordinates.**SolvateExistingStructure**(*protocol_id*) Solvates a set of 3D coordinates with a specified solvent using the PACKMOL package.

id	The unique id of this protocol.
mass_density	Input - The target density of the created system.
<pre>max_molecules</pre>	Input - The maximum number of molecules to be
	added to the system.
<pre>output_number_of_molecules</pre>	Output - The number of molecules in the created sys-
	tem.
output_substance	Output - The substance which was built by packmol.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
<pre>retain_packmol_files</pre>	Input - If True, packmol will not delete all of the tem-
	porary files it creates while building the coordinates.
schema	A serializable schema for this object.
solute_coordinate_file	Input - A file path to the solute to solvate.
substance	Input - The composition of the system to build.
tolerance	Input - The packmol distance tolerance in units com-
	patible with angstroms.
verbose_packmol	Input - If True, packmol will print verbose informa-
	tion to the logger The default value of this attribute is
	False.

Table 276 - continued from previous page

solute_coordinate_file

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

center_solute_in_box

Input - If *True*, the solute to solvate will be centered in the simulation box. The default value of this attribute is True.

Type bool

allow_merging

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 providied.

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

assigned_residue_names

Output - The residue names which were assigned to each of the components. Each key corresponds to a component identifier. The default value of this attribute is not set and must be set by the user.

Type dict

box_aspect_ratio

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** (Protocol) 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

Output - The file path to the created PDB coordinate file. The default value of this attribute is not set and must be set by the user..

Type str

count_exact_amount

Input - Whether components present in an exact amount (i.e. defined with an ExactAmount) should be considered when apply the maximum number of molecules constraint. This may be set false, for example, when building a separate solvated protein (n = 1) and solvated protein + ligand complex (n = 2) system but wish for both systems to have the same number of solvent molecules. 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

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(*file_path*)

Create this object from a JSON file.

Parameters file_path (str) - The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

mass_density

Input - The target density of the created system. The default value of this attribute is 0.95 g / ml.

Type Quantity

max_molecules

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 Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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

Output - The number of molecules in the created system. This may be less than maximum requested due to rounding of mole fractions The default value of this attribute is not set and must be set by the user.

Type int

output_substance

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. The default value of this attribute is not set and must be set by the user.

Type Substance

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

retain_packmol_files

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)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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

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

tolerance

Input - The pack mol distance tolerance in units compatible with angstroms. The default value of this attribute is 2.0 Å.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If attribute_type is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

verbose_packmol

Input - If True, packmol will print verbose information to the logger The default value of this attribute is False.

Type bool

BuildDockedCoordinates

class openff.evaluator.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)

Methods

__init__(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
can_merge(other[, path_replacements])	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
	continuos on novt nors

continues on next page

	contantaod nom providuo pago
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Table 277 - continued from previous page

Attributes

activate_site_location	Input - Defines the method by which the activate site
	is identified.
allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
<pre>docked_complex_coordinate_path</pre>	Output - The file path to the docked ligand-receptor
	complex.
docked_ligand_coordinate_path	Output - The file path to the coordinates of the lig-
	and in it's docked pose, aligned with the initial recep-
	tor_coordinate_file.
id	The unique id of this protocol.
ligand_residue_name	Output - The residue name assigned to the docked
	ligand.
ligand_substance	Input - A substance containing only the ligand to
	dock.
number_of_ligand_conformers	Input - The number of conformers to try and dock
	into the receptor structure.
outputs	A dictionary of the outputs of this property.
receptor_coordinate_file	Input - The file path to the MOL2 coordinates of the
	receptor molecule.
<pre>receptor_residue_name</pre>	Output - The residue name assigned to the receptor.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.

class ActivateSiteLocation(value)

An enum which describes the methods by which a receptors activate site(s) is located.

ligand_substance

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

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

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

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

Output - The file path to the coordinates of the ligand in it's docked pose, aligned with the initial *receptor_coordinate_file*. The default value of this attribute is not set and must be set by the user.

Type str

docked_complex_coordinate_path

Output - The file path to the docked ligand-receptor complex. The default value of this attribute is not set and must be set by the user.

Type str

ligand_residue_name

Output - The residue name assigned to the docked ligand. The default value of this attribute is not set and must be set by the user..

Type str

receptor_residue_name

Output - The residue name assigned to the receptor. The default value of this attribute is not set and must be set by the user.

Type str

allow_merging

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

Force Field Assignment

	set of force field parameters to a given system.
	set of force field parameters to a given system.
BaseBuildSystem	The base class for any protocol whose role is to apply a

continues on next page

Table 279 – continued nom previous page	
BuildSmirnoffSystem	Parametrise a set of molecules with a given smirnoff
	force field using the OpenFF toolkit.
BuildLigParGenSystem	Parametrise a set of molecules with the OPLS-AA/M
	force field.
BuildTLeapSystem	Parametrise a set of molecules with an Amber based
	force field.

Table 279 - continued from previous page

BaseBuildSystem

class openff.evaluator.protocols.forcefield.BaseBuildSystem(protocol_id)

The base class for any protocol whose role is to apply a set of force field parameters to a given system.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
coordinate_file_path	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.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
force_field_path	Input - The file path to the force field parameters to
	assign to the system.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
parameterized_system	Output - The parameterized system object.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
substance	Input - The composition of the system.

force_field_path

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

coordinate_file_path

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

substance

Input - The composition of the system. The default value of this attribute is not set and must be set by the user.

Type Substance

parameterized_system

Output - The parameterized system object. The default value of this attribute is not set and must be set by the user..

Type ParameterizedSystem

allow_merging

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 providied.

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

```
json(file_path=None, format=False)
```

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

BuildSmirnoffSystem

class openff.evaluator.protocols.forcefield.**BuildSmirnoffSystem**(*protocol_id*) Parametrise a set of molecules with a given smirnoff force field using the OpenFF toolkit.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
coordinate_file_path	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.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
force_field_path	Input - The file path to the force field parameters to
	assign to the system.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
parameterized_system	Output - The parameterized system object.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
substance	Input - The composition of the system.

allow_merging

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 providied.

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** (Protocol) 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

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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

force_field_path

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

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) - The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

parameterized_system

Output - The parameterized system object. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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

Input - The composition of the system. The default value of this attribute is not set and must be set by the user.

Type Substance

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

BuildLigParGenSystem

class openff.evaluator.protocols.forcefield.**BuildLigParGenSystem**(*protocol_id*) Parametrise a set of molecules with the OPLS-AA/M force field. using a LigParGen server.

Notes

This protocol is currently a work in progress and as such has limited functionality compared to the more established *BuildSmirnoffSystem* protocol.

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__(protocol_id)

Methods

___init___(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.

continues on next page

	1 10
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Table 284 – continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
coordinate_file_path	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.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
force_field_path	Input - The file path to the force field parameters to
	assign to the system.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
parameterized_system	Output - The parameterized system object.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
substance	Input - The composition of the system.
water_model	Input - The water model to apply, if any water
	molecules are present.

class WaterModel(value)

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

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 (*boo1*) – 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 providied.

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** (Protocol) 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

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

execute(*directory=''*, *available_resources=None*) Execute the protocol.

Parameters

• **directory** (*str*) – The directory to store output data in.

• **available_resources** (ComputeResources) – The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

force_field_path

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

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

parameterized_system

Output - The parameterized system object. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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

Input - The composition of the system. The default value of this attribute is not set and must be set by the user.

Type Substance

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

water_model

Input - The water model to apply, if any water molecules are present. The default value of this attribute is WaterModel.TIP3P.

Type TemplateBuildSystem.WaterModel

BuildTLeapSystem

```
class openff.evaluator.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)

Methods

___init__(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's, attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
charge_backend	Input - The backend framework to use to assign par-
	tial charges.
coordinate_file_path	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.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
force_field_path	Input - The file path to the force field parameters to
	assign to the system.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
parameterized_system	Output - The parameterized system object.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
substance	Input - The composition of the system.
water_model	Input - The water model to apply, if any water
	molecules are present.

class ChargeBackend(value)

The framework to use to assign partial charges.

charge_backend

Input - The backend framework to use to assign partial charges.

Type BuildTLeapSystem.ChargeBackend

class WaterModel(value)

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

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** (Protocol) 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

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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

force_field_path

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

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

parameterized_system

Output - The parameterized system object. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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

Input - The composition of the system. The default value of this attribute is not set and must be set by the user.

Type Substance

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

water_model

Input - The water model to apply, if any water molecules are present. The default value of this attribute is WaterModel.TIP3P.

Type TemplateBuildSystem.WaterModel

Gradients

ZeroGradients	Zeros the gradients of an observable with respect to a
	specified set of force field parameters.

ZeroGradients

class openff.evaluator.protocols.gradients.ZeroGradients(protocol_id)

Zeros the gradients of an observable with respect to a specified set of force field parameters.

__init__(protocol_id)

Methods

___init___(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

Input - Defines whether this protocols is allowed to
merge with other protocols.
A list of pointers to the protocols which this protocol
takes input from.
Input - The path to the force field which contains
the parameters to differentiate the observable with re-
spect to.
Input - The parameters to zero the gradient with re-
spect to.
The unique id of this protocol.
Input - The observable to set the gradients of.
Output - The observable with zeroed gradients.
A dictionary of the outputs of this property.
The inputs which must be set on this protocol.
A serializable schema for this object.

input_observables

Input - The observable to set the gradients of. The default value of this attribute is not set and must be set by the user.

Type typing.Union[*openff.evaluator.utils.observables.Observable*, *openff.evaluator.utils.observables.ObservableArray*]

force_field_path

Input - The path to the force field which contains the parameters to differentiate the observable with respect to. This is many used to get the correct units for the parameters. The default value of this attribute is not set and must be set by the user..

Type str

gradient_parameters

Input - The parameters to zero the gradient with respect to.

Type list

output_observables

Output - The observable with zeroed gradients. The default value of this attribute is not set and must be set by the user..

Type typing.Union[openff.evaluator.utils.observables.Observable, openff.evaluator.utils.observables.ObservableArray]

allow_merging

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

```
json(file_path=None, format=False)
```

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

Groups

ConditionalGroup	A collection of protocols which are to execute until a
	given condition is met.

ConditionalGroup

class openff.evaluator.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 ProtocolGroup.

Methods

init(protocol_id)	Constructs a new ProtocolGroup.
add_condition(condition_to_add)	Adds a condition to this groups list of conditions if it
	not already in the condition list.
add_protocols(*protocols)	Add protocols to this group.
apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another ProtocolGroup with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Store the uuid of the calculation this protocol belongs
	to
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to merge with other protocols.
conditions	Input - The conditions which must be satisfied be-
	forethe group will cleanly exit.
current_iteration	Output - The current number of iterations this group
	has performed while attempting to satisfy the speci-
	fied conditions.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.

continues on next page

	ee eentinded nom providuo page
id	The unique id of this protocol.
max_iterations	Input - The maximum number of iterations to run for
	to try and satisfy the groups conditions.
outputs	A dictionary of the outputs of this property.
protocols	A dictionary of the protocols in this groups, where
	the dictionary key is the protocol id, and the value is
	the protocol itself.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.

Table	293 –	continued	from	previous	page
-------	-------	-----------	------	----------	------

class Condition

Defines a specific condition which must be met of the form *left_hand_value* [TYPE] *right_hand_value*, where *[TYPE]* may be less than or greater than.

class Type(value)

The available condition types.

left_hand_value

The left-hand value to compare. The default value of this attribute is not set and must be set by the user..

Type typing.Union[int, float, openff.evaluator.utils.units.Quantity]

right_hand_value

The right-hand value to compare. The default value of this attribute is not set and must be set by the user.

Type typing.Union[int, float, openff.evaluator.utils.units.Quantity]

type

The right-hand value to compare. The default value of this attribute is Type.LessThan. Type ConditionalGroup.Condition.Type

classmethod from_json(file_path)

Create this object from a JSON file. **Parameters file_path** (*str*) – The path to load the JSON from. **Returns** The parsed class. **Return type** cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type. **Return type** list of str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

• file_path (str, optional) – The (optional) file path to save the JSON file to.

• **format** (*bool*) – Whether to format the JSON or not.

- **Returns** The JSON representation of this class.
- Return type str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string. **Returns** The parsed class.

Return type Any

validate(attribute_type=None)

Validate the values of the attributes. If attribute_type is set, only attributes of that type will be validated.
Parameters attribute_type (type of Attribute, optional) – The type of attribute
to validate.

Raises ValueError or AssertionError -

conditions

Input - The conditions which must be satisfied before the group will cleanly exit. The default value of this attribute is [].

Type list

current_iteration

Output - The current number of iterations this group has performed while attempting to satisfy the specified conditions. This value starts from one. The default value of this attribute is not set and must be set by the user.

Type int

max_iterations

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

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.

get_value_references(input_path)

Returns a dictionary of references to the protocols which one of this protocols inputs (specified by *in-put_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 (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

add_protocols(*protocols)

Add protocols to this group.

Parameters protocols (Protocol) – The protocols to add.

allow_merging

Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is **True**.

Type bool

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 providied.

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** (Protocol) 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

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

```
json(file_path=None, format=False)
Creates a JSON representation of this class.
```

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

property protocols

A dictionary of the protocols in this groups, where the dictionary key is the protocol id, and the value is the protocol itself.

Notes

This property should not be altered. Use add_protocols to add new protocols to the group.

Type dict of str and Protocol

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.

property required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

Miscellaneous

AddValues	A protocol to add together a list of values.
SubtractValues	A protocol to subtract one value from another such that:
MultiplyValue	A protocol which multiplies a value by a specified scalar
DivideValue	A protocol which divides a value by a specified scalar
WeightByMoleFraction	Multiplies a value by the mole fraction of a component
	in a Substance.
FilterSubstanceByRole	A protocol which takes a substance as input, and returns
	a substance which only contains components whose role
	match a given criteria.
DummyProtocol	A protocol whose only purpose is to return an input value
	as an output value.

AddValues

class openff.evaluator.protocols.miscellaneous.**AddValues**(*protocol_id*) A protocol to add together a list of values.

Notes

The *values* input must either be a list of openff.evaluator.unit.Quantity, a ProtocolPath to a list of openff.evaluator.unit.Quantity, or a list of ProtocolPath which each point to a openff.evaluator.unit.Quantity.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
	continues on next page

	1 1 0
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
set_value(reference_path, value)	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Table	295 –	continued	from	previous	page
-------	-------	-----------	------	----------	------

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
result	Output - The sum of the values.
schema	A serializable schema for this object.
values	Input - The values to add together.

values

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

Output - The sum of the values. The default value of this attribute is not set and must be set by the user.

Type typing.Union[int, float, openff.evaluator.utils.units.Measurement, openff.evaluator.utils.units.Quantity, openff.evaluator.forcefield.gradients.ParameterGradient, openff.evaluator.utils.observables.Observable, openff.evaluator.utils.observables.ObservableArray]

allow_merging

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 providied.

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

SubtractValues

class openff.evaluator.protocols.miscellaneous.**SubtractValues**(*protocol_id*) A protocol to subtract one value from another such that:

 $result = value_b - value_a$

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
from_json(file_path)	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
	continues on next page

result	Output - The results of <i>value_b</i> - <i>value_a</i> .
schema	A serializable schema for this object.
value_a	Input - value_a in the formula result = value_b -
	value_a.
value_b	Input - value_b in the formula result = value_b -
	value_a.

Table 298 - continued from previous page

value_a

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.

Typetyping.Union[int,float,openff.evaluator.utils.units.Measurement,openff.evaluator.utils.units.Quantity, openff.evaluator.forcefield.gradients.ParameterGradient,openff.evaluator.utils.observables.Observable, openff.evaluator.utils.observables.ObservableArray]

value_b

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.

Typetyping.Union[int,float,openff.evaluator.utils.units.Measurement,openff.evaluator.utils.units.Quantity, openff.evaluator.forcefield.gradients.ParameterGradient,openff.evaluator.utils.observables.Observable, openff.evaluator.utils.observables.ObservableArray]

result

Output - The results of *value_b* - *value_a*. The default value of this attribute is not set and must be set by the user.

Typetyping.Union[int,float,openff.evaluator.utils.units.Measurement,openff.evaluator.utils.units.Quantity, openff.evaluator.forcefield.gradients.ParameterGradient,openff.evaluator.utils.observables.Observable, openff.evaluator.utils.observables.ObservableArray]

allow_merging

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 (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None) Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

```
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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

MultiplyValue

class openff.evaluator.protocols.miscellaneous.**MultiplyValue**(*protocol_id*) A protocol which multiplies a value by a specified scalar

__init__(protocol_id)

Methods

__init__(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
from_json(file_path)	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
multiplier	Input - The scalar to multiply by.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
result	Output - The result of the multiplication.
schema	A serializable schema for this object.
value	Input - The value to multiply.

value

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, openff.evaluator.utils.units.Measurement, openff.evaluator.utils.units.Quantity, openff.evaluator.forcefield.gradients.ParameterGradient, openff.evaluator.utils.observables.Observable, openff.evaluator.utils.observables.ObservableArray]

multiplier

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, openff.evaluator.utils.units.Quantity]

result

Output - The result of the multiplication. The default value of this attribute is not set and must be set by the user.

Type typing.Union[int, float, openff.evaluator.utils.units.Measurement, openff.evaluator.utils.units.Quantity, openff.evaluator.forcefield.gradients.ParameterGradient, openff.evaluator.utils.observables.Observable, openff.evaluator.utils.observables.ObservableArray]

allow_merging

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- directory (str) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(*file_path=None*, *format=False*)

Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

```
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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If attribute_type is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

DivideValue

class openff.evaluator.protocols.miscellaneous.**DivideValue**(*protocol_id*) A protocol which divides a value by a specified scalar

__init__(protocol_id)

Methods

___init__(protocol_id)

Applies a <i>ProtocolReplicator</i> to this protocol.
Determines whether this protocol can be merged with
another.
Execute the protocol.
Create this object from a JSON file.
Initializes a protocol from it's schema definition.
Returns all attributes of a specific <i>attribute_type</i> .
Returns one of this protocols, or any of its children's,
attributes directly (rather than its value).
Returns the value of one of this protocols inputs / out-
puts.
Returns a dictionary of references to the protocols
which one of this protocols inputs (specified by in-
<i>put_path</i>) takes its value from.
Creates a JSON representation of this class.
Merges another Protocol with this one.
Parses a typed json string into the corresponding class
structure.
Finds each input which came from a given protocol
Prepend a unique identifier to this protocols id.
~
Sets the value of one of this protocols inputs. Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
divisor	Input - The scalar to divide by.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
result	Output - The result of the division.
schema	A serializable schema for this object.
value	Input - The value to divide.

value

Input - The value to divide. The default value of this attribute is not set and must be set by the user.

Typetyping.Union[int,float,openff.evaluator.utils.units.Measurement,openff.evaluator.utils.units.Quantity, openff.evaluator.forcefield.gradients.ParameterGradient,openff.evaluator.utils.observables.Observable, openff.evaluator.utils.observables.ObservableArray]

divisor

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, openff.evaluator.utils.units.Quantity]

result

Output - The result of the division. The default value of this attribute is not set and must be set by the user.

 Type
 typing.Union[int,
 float,
 openff.evaluator.utils.units.Measurement,

 openff.evaluator.utils.units.Quantity, openff.evaluator.forcefield.gradients.ParameterGradient,
 openff.evaluator.utils.observables.Observable, openff.evaluator.utils.observables.Observable, openff.evaluator.utils.observables.ObservableArray]

allow_merging

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** (*boo1*) – 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 providied.

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** (Protocol) 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

execute(*directory=''*, *available_resources=None*) Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(*file_path=None*, *format=False*)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

WeightByMoleFraction

class openff.evaluator.protocols.miscellaneous.WeightByMoleFraction(*protocol_id*) Multiplies a value by the mole fraction of a component in a *Substance*.

__init__(protocol_id)

Methods

 __init__(protocol_id)

 apply_replicator(replicator, template_values)
 Applies a ProtocolReplicator to this protocol.

 can_merge(other[, path_replacements])
 Determines whether this protocol can be merged with another.

 execute([directory, available_resources])
 Execute the protocol.

 from_json(file_path)
 Create this object from a JSON file.

 continues on next page

	ded from previous page
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Table 303 - continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
component	Input - The component whose mole fraction to
	weight by.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
full_substance	Input - The full substance which describes the mole
	fraction of the component.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
value	Input - The value to be weighted.
weighted_value	Output - The value weighted by the <i>component</i> 's
	mole fraction as determined from the `full_substance.

value

Input - The value to be weighted. The default value of this attribute is not set and must be set by the user..

Typetyping.Union[int,float,openff.evaluator.utils.units.Measurement,openff.evaluator.utils.units.Quantity, openff.evaluator.forcefield.gradients.ParameterGradient,openff.evaluator.utils.observables.Observable, openff.evaluator.utils.observables.ObservableArray]

component

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

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

Output - The value weighted by the *component*'s mole fraction as determined from the 'full_substance. The default value of this attribute is not set and must be set by the user.

 Type
 typing.Union[int,
 float,
 openff.evaluator.utils.units.Measurement,

 openff.evaluator.utils.units.Quantity, openff.evaluator.forcefield.gradients.ParameterGradient,

 $open {\it ff.evaluator.utils.observables.Observable,open {\it ff.evaluator.utils.observables.ObservableArray}]$

allow_merging

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** (Protocol) – 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

FilterSubstanceByRole

class openff.evaluator.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)

Methods

___init___(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
component_roles	Input - The roles to filter substance components
	against.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
expected_components	Input - The number of components expected to re-
	main after filtering.
filtered_substance	Output - The filtered substance.
id	The unique id of this protocol.
input_substance	Input - The substance to filter.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.

input_substance

Input - The substance to filter. The default value of this attribute is not set and must be set by the user.

Type Substance

component_roles

Input - The roles to filter substance components against. The default value of this attribute is not set and

must be set by the user ..

Type list

expected_components

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 attribute is *optional*.

Type int

filtered_substance

Output - The filtered substance. The default value of this attribute is not set and must be set by the user.

Type Substance

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

allow_merging

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** (*boo1*) – 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 providied.

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

DummyProtocol

class openff.evaluator.protocols.miscellaneous.DummyProtocol(protocol_id)

A protocol whose only purpose is to return an input value as an output value.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
input_value	Input - A dummy input.
output_value	Output - A dummy output.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.

input_value

Input - A dummy input. The default value of this attribute is not set and must be set by the user..

 Type
 typing.Union[str,
 int,
 float,
 openff.evaluator.utils.units.Quantity,

 openff.evaluator.utils.units.Measurement,
 openff.evaluator.utils.observables.Observable,
 openff.evaluator.utils.observables.Observable,

 openff.evaluator.utils.observables.Observables.ObservableArray, openff.evaluator.forcefield.gradients.ParameterGradient,
 openff.evaluator.forcefield.gradients.ParameterGradientKey,

output_value

Output - A dummy output. The default value of this attribute is not set and must be set by the user..

Type typing.Union[str, int, float, openff.evaluator.utils.units.Quantity,

openff.evaluator.utils.units.Measurement, openff.evaluator.utils.observables.Observable, openff.evaluator.utils.observables.ObservableArray, openff.evaluator.forcefield.gradients.ParameterGradient, openff.evaluator.forcefield.gradients.ParameterGradientKey, list, tuple, dict, set, frozenset]

allow_merging

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 providied.

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

```
Return type cls
```

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

OpenMM

OpenMMEnergyMinimisation	A protocol to minimise the potential energy of a system using OpenMM.
OpenMMSimulation	Performs a molecular dynamics simulation in a given en- semble using an OpenMM backend.
<i>OpenMMEvaluateEnergies</i>	Re-evaluates the energy of a series of configurations for a given set of force field parameters using OpenMM.

OpenMMEnergyMinimisation

class openff.evaluator.protocols.openmm.**OpenMMEnergyMinimisation**(*protocol_id*) A protocol to minimise the potential energy of a system using OpenMM.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
enable_pbc	Input - If true, periodic boundary conditions will be
	enabled.
id	The unique id of this protocol.
<pre>input_coordinate_file</pre>	Input - The coordinates to minimise.
<pre>max_iterations</pre>	Input - The maximum number of iterations to per-
	form.
<pre>output_coordinate_file</pre>	Output - The file path to the minimised coordinates.
outputs	A dictionary of the outputs of this property.
	continues on peyt page

continues on next page

Table of the continued from previous page	
parameterized_system	Input - The parameterized system object which en-
	codes the systems potential energy function.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
tolerance	Input - The energy tolerance to which the system
	should be minimized.

Table 311 - continued from previous pa	ge
--	----

allow_merging

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** (Protocol) – 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

enable_pbc

Input - If true, periodic boundary conditions will be enabled. The default value of this attribute is True.

Type bool

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

input_coordinate_file

Input - The coordinates to minimise. The default value of this attribute is not set and must be set by the user..

Type str

```
json(file_path=None, format=False)
```

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

max_iterations

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

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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_coordinate_file

Output - The file path to the minimised coordinates. The default value of this attribute is not set and must be set by the user.

Type str

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

parameterized_system

Input - The parameterized system object which encodes the systems potential energy function. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

tolerance

Input - The energy tolerance to which the system should be minimized. The default value of this attribute is 10.0 kJ / mol.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

OpenMMSimulation

class openff.evaluator.protocols.openmm.OpenMMSimulation(protocol_id)

Performs a molecular dynamics simulation in a given ensemble using an OpenMM backend.

This protocol employs the Langevin integrator implemented in the openmntools package to propagate the state of the system using the default BAOAB splitting [1]_. Further, simulations which are run in the NPT simulation will have a Monte Carlo barostat (simtk.openmm.MonteCarloBarostat) applied every 25 steps (the OpenMM default).

References

[1] Leimkuhler, Ben, and Charles Matthews. "Numerical methods for stochastic molecular dynamics." Molecular Dynamics. Springer, Cham, 2015. 261-328.

__init__(protocol_id)

Methods

__init__(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
<pre>get_value(reference_path)</pre>	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
	continues on next page

Table	312 – continued from previous page

<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

allow_gpu_platforms	Input - If true, the simulation will be performed
	using a GPU if available, otherwise it will be con-
	strained to only using CPUs.
allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
checkpoint_frequency	Input - The frequency (in multiples of <i>out</i> -
	<i>put_frequency</i>) with which to write to a checkpoint
Annual and a second	file, e.g.
dependencies	A list of pointers to the protocols which this protocol
anahla aha	takes input from.
enable_pbc	Input - If true, periodic boundary conditions will be
1 1	enabled.
ensemble	Input - The thermodynamic ensemble to simulate in.
gradient_parameters	Input - An optional list of parameters to differentiate
	the evaluated energies with respect to.
high_precision	Input - If true, the simulation will be run using dou-
	ble precision.
id	The unique id of this protocol.
<pre>input_coordinate_file</pre>	Input - The file path to the starting coordinates.
observables	Output - The observables collected during the sim-
	ulation.
output_coordinate_file	Output - The file path to the coordinates of the final
	system configuration.
output_frequency	Input - The frequency (in number of steps) with
	which to write to the output statistics and trajectory
	files.
outputs	A dictionary of the outputs of this property.
parameterized_system	Input - The parameterized system object which en-
	codes the systems potential energy function.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
steps_per_iteration	Input - The number of steps to propogate the system
	by at each iteration.
thermodynamic_state	Input - The thermodynamic conditions to simulate
	under The default value of this attribute is not set and
	must be set by the user
thermostat_friction	Input - The thermostat friction coefficient.
timestep	Input - The timestep to evolve the system by at each
	step.
total_number_of_iterations	Input - The number of times to propogate the system
total_number_of_iterations	Input - The number of times to propogate the system forward by the <i>steps_per_iteration</i> number of steps.
<pre>total_number_of_iterations trajectory_file_path</pre>	Input - The number of times to propogate the system

allow_gpu_platforms

Input - If true, the simulation will be performed using a GPU if available, otherwise it will be constrained

to only using CPUs. The default value of this attribute is True.

Type bool

allow_merging

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 (Protocol) 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_frequency

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. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 10. This attribute is *optional*.

Type int

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

enable_pbc

Input - If true, periodic boundary conditions will be enabled. The default value of this attribute is True.

Type bool

ensemble

Input - The thermodynamic ensemble to simulate in. The default value of this attribute is Ensemble.NPT.

Type Ensemble

execute(directory=", available_resources=None)
Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(*attribute_type=None*) Returns all attributes of a specific *attribute type*.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

gradient_parameters

Input - An optional list of parameters to differentiate the evaluated energies with respect to.

Type list

high_precision

Input - If true, the simulation will be run using double precision. The default value of this attribute is False.

Type bool

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

input_coordinate_file

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

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

observables

Output - The observables collected during the simulation. The default value of this attribute is not set and must be set by the user.

Type ObservableFrame

output_coordinate_file

Output - The file path to the coordinates of the final system configuration. The default value of this attribute is not set and must be set by the user.

Type str

output_frequency

Input - The frequency (in number of steps) with which to write to the output statistics and trajectory files. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 3000.

Type int

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

parameterized_system

Input - The parameterized system object which encodes the systems potential energy function. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

steps_per_iteration

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

thermodynamic_state

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

thermostat_friction

Input - The thermostat friction coefficient. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 1.0 / ps.

Type Quantity

timestep

Input - The timestep to evolve the system by at each step. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 2.0 fs.

Type Quantity

total_number_of_iterations

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

trajectory_file_path

Output - The file path to the trajectory sampled during the simulation. The default value of this attribute is not set and must be set by the user.

Type str

validate(attribute_type=None)

Validate the values of the attribute. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

OpenMMEvaluateEnergies

```
class openff.evaluator.protocols.openmm.OpenMMEvaluateEnergies(protocol_id)
```

Re-evaluates the energy of a series of configurations for a given set of force field parameters using OpenMM.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

merge with other protocols.A list of pointers to the protocols which this protocol takes input from.Input - If true, periodic boundary conditions will be enabled.Input - An optional list of parameters to differentiate the evaluated energies with respect to.
takes input from.Input - If true, periodic boundary conditions will be enabled.Input - An optional list of parameters to differentiate
Input - If true, periodic boundary conditions will be enabled.Input - An optional list of parameters to differentiate
enabled. Input - An optional list of parameters to differentiate
Input - An optional list of parameters to differentiate
the evaluated energies with respect to
the evaluated chergies with respect to.
The unique id of this protocol.
Output - An observable array which stores the re-
duced potentials potential energies evaluated at the
specified state and using the specified system object
for each configuration in the trajectory.
continues on next page

	Table ere continued nem providue page
outputs	A dictionary of the outputs of this property.
parameterized_system	Input - The parameterized system object which en-
	codes the systems potential energy function.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
thermodynamic_state	Input - The state to calculate the reduced potentials
	at.
<pre>trajectory_file_path</pre>	Input - The path to the trajectory file which contains
	the configurations to calculate the energies of.

Table 315 - continued from previous page

allow_merging

Input - Defines whether this protocols is allowed to merge with other protocols. The default value of this attribute is **True**.

Type bool

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 providied.

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** (Protocol) 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

enable_pbc

Input - If true, periodic boundary conditions will be enabled. The default value of this attribute is True.

Type bool

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

gradient_parameters

Input - An optional list of parameters to differentiate the evaluated energies with respect to.

Type list

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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_observables

Output - An observable array which stores the reduced potentials potential energies evaluated at the specified state and using the specified system object for each configuration in the trajectory. The default value of this attribute is not set and must be set by the user.

Type ObservableFrame

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

parameterized_system

Input - The parameterized system object which encodes the systems potential energy function. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

thermodynamic_state

Input - The state to calculate the reduced potentials at. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

trajectory_file_path

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

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

Paprika

PreparePullCoordinates	A protocol which will align a host-guest complex to the
	z-axis and position the guest molecule at a specified
	point along the pull axis.
PrepareReleaseCoordinates	A protocol which will extract the host molecule from
	a file containing both the host and guest molecules and
	produce a coordinate file containing only the host which
	has been correctly aligned to the z-axis.
AddDummyAtoms	A protocol which will add the reference 'dummy' atoms
	to a parameterised system.

PreparePullCoordinates

class openff.evaluator.protocols.paprika.coordinates.**PreparePullCoordinates**(*protocol_id*) A protocol which will align a host-guest complex to the z-axis and position the guest molecule at a specified point along the pull axis.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
can_merge(other[, path_replacements])	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.

continues on next page

	continued norm previous page
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Table 317 - continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
complex_file_path	Input - The path to the file which the coordinates of
	the guest moleculebound to the host molecule.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
guest_orientation_mask	Input - The string mask which describes which guest
	atoms will be restrained relative to the dummy atoms
	to keep the molecule aligned to the z-axis.
id	The unique id of this protocol.
n_pull_windows	Input - The total number of the pull windows in the
	calculation.
output_coordinate_path	Output - The file path to the system which has been
	correctly aligned to the z-axis.
outputs	A dictionary of the outputs of this property.
pull_distance	Input - The total distance that the guest will be pulled
	along the z-axis during the pull phase.
pull_window_index	Input - The index of the pull window to generate co-
	ordinates for.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
substance	Input - The substance which defines the host, guest
	and solvent.
-	

guest_orientation_mask

Input - The string mask which describes which guest atoms will be restrained relative to the dummy atoms to keep the molecule aligned to the z-axis. This should be of the form 'X Y' where X Y are ParmEd selectors for the first and second guest atoms. The default value of this attribute is not set and must be set by the user..

Type str

pull_distance

Input - The total distance that the guest will be pulled along the z-axis during the pull phase. The default value of this attribute is not set and must be set by the user.

Type Quantity

pull_window_index

Input - The index of the pull window to generate coordinates for. The default value of this attribute is not set and must be set by the user.

Type int

n_pull_windows

Input - The total number of the pull windows in the calculation. The default value of this attribute is not

set and must be set by the user..

Type int

allow_merging

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 providied.

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 (Protocol) 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

complex_file_path

Input - The path to the file which the coordinates of the guest moleculebound to the host molecule. 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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_coordinate_path

Output - The file path to the system which has been correctly aligned to the z-axis. The default value of this attribute is not set and must be set by the user.

Type str

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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

Input - The substance which defines the host, guest and solvent. The default value of this attribute is not set and must be set by the user.

Type Substance

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

PrepareReleaseCoordinates

class openff.evaluator.protocols.paprika.coordinates.**PrepareReleaseCoordinates**(*protocol_id*) A protocol which will extract the host molecule from a file containing both the host and guest molecules and produce a coordinate file containing only the host which has been correctly aligned to the z-axis.

__init__(protocol_id)

Methods

__init__(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
complex_file_path	Input - The path to the file which the coordinates of
	the guest moleculebound to the host molecule.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
output_coordinate_path	Output - The file path to the system which has been
	correctly aligned to the z-axis.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
	continues on port page

continues on next page

Table 320 – continued from previous page	Table	320 -	 continued 	from	previous page
--	-------	-------	-------------------------------	------	---------------

	· · · · ·
substance	Input - The substance which defines the host, guest
	and solvent.

allow_merging

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** (Protocol) 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

complex_file_path

Input - The path to the file which the coordinates of the guest moleculebound to the host molecule. 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- directory (str) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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_coordinate_path

Output - The file path to the system which has been correctly aligned to the z-axis. The default value of this attribute is not set and must be set by the user.

Type str

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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

Input - The substance which defines the host, guest and solvent. The default value of this attribute is not set and must be set by the user.

Type Substance

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

AddDummyAtoms

class openff.evaluator.protocols.paprika.coordinates.AddDummyAtoms(protocol_id)

A protocol which will add the reference 'dummy' atoms to a parameterised system. This protocol assumes the host / complex has already been correctly aligned to the z-axis and has been placed at the origin.

__init__(protocol_id)

Methods

__init__(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_j son(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
input_coordinate_path	Input - The file path to the coordinates which the
	dummy atoms should be added to.
input_system	Input - The parameterized system which the dummy
	atoms should be added to.
offset	Input - The distance to offset the dummy atoms from
	the origin $(0, 0, 0)$ backwards along the z-axis.
	continues on next page

	TUDIC OLL	continued norm previous page
<pre>output_coordinate_path</pre>		Output - The file path to the coordinates which in-
		clude the added dummy atoms.
output_system		Output - The parameterized system which include
		the added dummy atoms.
outputs		A dictionary of the outputs of this property.
required_inputs		The inputs which must be set on this protocol.
schema		A serializable schema for this object.
substance		Input - The substance which defines the host, guest
		and solvent.

Table 322 - continued from previous page

substance

Input - The substance which defines the host, guest and solvent. The default value of this attribute is not set and must be set by the user.

Type Substance

offset

Input - The distance to offset the dummy atoms from the origin (0, 0, 0) backwards along the z-axis. The default value of this attribute is not set and must be set by the user.

Type Quantity

input_coordinate_path

Input - The file path to the coordinates which the dummy atoms should be added to. The default value of this attribute is not set and must be set by the user.

Type str

input_system

Input - The parameterized system which the dummy atoms should be added to. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

output_coordinate_path

Output - The file path to the coordinates which include the added dummy atoms. The default value of this attribute is not set and must be set by the user.

Type str

output_system

Output - The parameterized system which include the added dummy atoms. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

allow_merging

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 (*boo1*) – 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 providied.

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

GenerateAttachRestraints	Generates the restraint values to apply during the 'at-
Generater tacinces traines	
	tach' phase from a set of restraint schema definitions
	and makes them easily accessible for the protocols which
	will apply them to the parameterized system.
GeneratePullRestraints	Generates the restraint values to apply during the 'pull'
	phase from a set of restraint schema definitions and
	makes them easily accessible for the protocols which
	will apply them to the parameterized system.
GenerateReleaseRestraints	Generates the restraint values to apply during the 're-
	lease' phase from a set of restraint schema definitions
	and makes them easily accessible for the protocols which
	will apply them to the parameterized system.
ApplyRestraints	A protocol which will apply the restraints defined in a
	restraints JSON file to a specified system.

GenerateAttachRestraints

class openff.evaluator.protocols.paprika.restraints.**GenerateAttachRestraints**(*protocol_id*) Generates the restraint values to apply during the 'attach' phase from a set of restraint schema definitions and makes them easily accessible for the protocols which will apply them to the parameterized system.

__init__(protocol_id)

Methods

init(protocol_id)	
apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.

continues on next page

<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Table 324 – continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
attach_lambdas	Input - The values of lambda to use for the attach
	phase.
complex_coordinate_path	Input - The file path to a coordinate file which con-
	tains the solvatedhost-guest complex and has the an-
	chor dummy atoms added.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
restraint_schemas	Input - The full set of restraint schemas.
restraints_path	Output - The file path to the <i>paprika</i> generated re-
	straints JSON file.
schema	A serializable schema for this object.

complex_coordinate_path

Input - The file path to a coordinate file which contains the solvatedhost-guest complex and has the anchor dummy atoms added. The default value of this attribute is not set and must be set by the user.

Type str

attach_lambdas

Input - The values of lambda to use for the attach phase. These muststart from 0.0 and increase monotonically to and include 1.0. The default value of this attribute is not set and must be set by the user.

Type list

allow_merging

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** (*boo1*) – 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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) - The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

restraint_schemas

Input - The full set of restraint schemas. The default value of this attribute is not set and must be set by the user..

Type dict

restraints_path

Output - The file path to the *paprika* generated restraints JSON file. The default value of this attribute is not set and must be set by the user.

Type str

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended uuid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

validate(attribute type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

GeneratePullRestraints

class openff.evaluator.protocols.paprika.restraints.GeneratePullRestraints(protocol_id) Generates the restraint values to apply during the 'pull' phase from a set of restraint schema definitions and makes them easily accessible for the protocols which will apply them to the parameterized system.

__init__(protocol_id)

Methods

___init___(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
<pre>get_value(reference_path)</pre>	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
	continues on next page

continues on next page

json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Table 326 - continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
attach_lambdas	Input - The values of lambda to use for the attach
	phase.
<pre>complex_coordinate_path</pre>	Input - The file path to a coordinate file which con-
	tains the solvatedhost-guest complex and has the an-
	chor dummy atoms added.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
n_pull_windows	Input - The number of lambda to use for the pull
	phase.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
restraint_schemas	Input - The full set of restraint schemas.
restraints_path	Output - The file path to the paprika generated re-
	straints JSON file.
schema	A serializable schema for this object.

n_pull_windows

Input - The number of lambda to use for the pull phase. The default value of this attribute is not set and must be set by the user.

Type int

allow_merging

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 providied.

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

attach_lambdas

Input - The values of lambda to use for the attach phase. These muststart from 0.0 and increase monotonically to and include 1.0. The default value of this attribute is not set and must be set by the user.

Type list

can_merge(other, path_replacements=None)

Determines whether this protocol can be merged with another.

Parameters

- **other** (Protocol) 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

complex_coordinate_path

Input - The file path to a coordinate file which contains the solvatedhost-guest complex and has the anchor dummy atoms added. 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

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

restraint_schemas

Input - The full set of restraint schemas. The default value of this attribute is not set and must be set by the user.

Type dict

restraints_path

Output - The file path to the *paprika* generated restraints JSON file. The default value of this attribute is not set and must be set by the user.

Type str

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

GenerateReleaseRestraints

class openff.evaluator.protocols.paprika.restraints.GenerateReleaseRestraints(protocol_id)
Generates the restraint values to apply during the 'release' phase from a set of restraint schema definitions and
makes them easily accessible for the protocols which will apply them to the parameterized system.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
host_coordinate_path	Input - The file path to a coordinate file which con-
	tains the solvatedhost molecule and has the anchor
	dummy atoms added.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
release_lambdas	Input - The values of lambda to use for the release
	phase.
required_inputs	The inputs which must be set on this protocol.
	continues on next page

	habie eze contanded nom provided page
restraint_schemas	Input - The full set of restraint schemas.
restraints_path	Output - The file path to the <i>paprika</i> generated re- straints JSON file.
schema	A serializable schema for this object.

Table 329 - continued from previous page

host_coordinate_path

Input - The file path to a coordinate file which contains the solvatedhost molecule and has the anchor dummy atoms added. The default value of this attribute is not set and must be set by the user.

Type str

release_lambdas

Input - The values of lambda to use for the release phase. These must start from 1.0 and decrease monotonically to and include 0.0. The default value of this attribute is not set and must be set by the user.

Type list

allow_merging

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 providied.

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

```
json(file_path=None, format=False)
```

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

restraint_schemas

Input - The full set of restraint schemas. The default value of this attribute is not set and must be set by the user.

Type dict

restraints_path

Output - The file path to the *paprika* generated restraints JSON file. The default value of this attribute is not set and must be set by the user.

Type str

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

ApplyRestraints

class openff.evaluator.protocols.paprika.restraints.**ApplyRestraints**(*protocol_id*) A protocol which will apply the restraints defined in a restraints JSON file to a specified system.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
load_restraints(file_path)	Loads a set of <i>paprika</i> restraint objects from a JSON
	file.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.
validate([attribute type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
input_system	Input - The parameterized system which the re-
	straints should be added to.
output_system	Output - The parameterized system which now in-
	cludes the added restraints.
outputs	A dictionary of the outputs of this property.
phase	Input - The APR phase to take the restraints from.
	continues on next page

1001	le de l'étaillada nom provodo page
required_inputs	The inputs which must be set on this protocol.
restraints_path	Input - The file path to the JSON file which contains
	the restraint definitions.
schema	A serializable schema for this object.
window_index	Input - The index of the window to take the restraints
	from.

Table 331 – continued from previous page
--

restraints_path

Input - The file path to the JSON file which contains the restraint definitions. This will usually have been generated by a *GenerateXXXRestraints* protocol. The default value of this attribute is not set and must be set by the user..

Type str

phase

Input - The APR phase to take the restraints from. The default value of this attribute is not set and must be set by the user.

Type str

window_index

Input - The index of the window to take the restraints from. The default value of this attribute is not set and must be set by the user.

Type int

input_system

Input - The parameterized system which the restraints should be added to. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

output_system

Output - The parameterized system which now includes the added restraints. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

classmethod load_restraints(file_path: str)

Loads a set of *paprika* restraint objects from a JSON file.

Parameters file_path – The path to the JSON serialized restraints.

Returns

Return type The loaded *paprika* restraint objects.

allow_merging

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 providied.

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

AnalyzeAPRPhase	A protocol which will analyze the outputs of the attach, pull or release phases of an APR calculation and return the change in free energy for that phase of the calcula-
	tion.
ComputeSymmetryCorrection	Computes the symmetry correction for an APR calcula-
	tion which involves a guest with symmetry.
ComputeReferenceWork	Computes the reference state work.

AnalyzeAPRPhase

class openff.evaluator.protocols.paprika.analysis.AnalyzeAPRPhase(protocol_id)

A protocol which will analyze the outputs of the attach, pull or release phases of an APR calculation and return the change in free energy for that phase of the calculation.

__init__(protocol_id)

Methods

__init__(protocol_id)

Applies a <i>ProtocolReplicator</i> to this protocol.
Determines whether this protocol can be merged with
another.
Execute the protocol.
Create this object from a JSON file.
Initializes a protocol from it's schema definition.
Returns all attributes of a specific <i>attribute_type</i> .
Returns one of this protocols, or any of its children's,
attributes directly (rather than its value).
Returns the value of one of this protocols inputs / out-
puts.
Returns a dictionary of references to the protocols
which one of this protocols inputs (specified by in-
<i>put_path</i>) takes its value from.
Creates a JSON representation of this class.
Merges another Protocol with this one.
Parses a typed json string into the corresponding class
structure.
Finds each input which came from a given protocol
continues on next page

Table 335 – continued nom previous page	
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Table 333 - continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
phase	Input - The phase of the calculation being analyzed.
required_inputs	The inputs which must be set on this protocol.
restraints_path	Input - The file path to the JSON file which contains
	the restraint definitions.
result	Output - The analysed free energy.
schema	A serializable schema for this object.
topology_path	Input - The file path to a coordinate file which con-
	tains topological information about the system.
trajectory_paths	Input - A list of paths to the trajectories (in the cor-
	rect order) generated during the phase being ana-
	lyzed.

topology_path

Input - The file path to a coordinate file which contains topological information about the system. The default value of this attribute is not set and must be set by the user.

Type str

trajectory_paths

Input - A list of paths to the trajectories (in the correct order) generated during the phase being analyzed. The default value of this attribute is not set and must be set by the user.

Type list

phase

Input - The phase of the calculation being analyzed. The default value of this attribute is not set and must be set by the user.

Type str

restraints_path

Input - The file path to the JSON file which contains the restraint definitions. This will usually have been generated by a *GenerateXXXRestraints* protocol. The default value of this attribute is not set and must be set by the user.

Type str

result

Output - The analysed free energy. The default value of this attribute is not set and must be set by the user..

Type Observable

allow_merging

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** (Protocol) 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

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(*file_path*)

Create this object from a JSON file.

Parameters file_path (str) - The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

ComputeSymmetryCorrection

class openff.evaluator.protocols.paprika.analysis.**ComputeSymmetryCorrection**(*protocol_id*) Computes the symmetry correction for an APR calculation which involves a guest with symmetry.

__init__(protocol_id)

Methods

 __init__(protocol_id)

 apply_replicator(replicator, template_values)
 Applies a ProtocolReplicator to this protocol.

 can_merge(other[, path_replacements])
 Determines whether this protocol can be merged with another.

 execute([directory, available_resources])
 Execute the protocol.

 from_json(file_path)
 Create this object from a JSON file.

 continues on next page

	ea nem previeue page
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
set_value(reference_path, value)	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Table 335 – continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
n_microstates	Input - The number of symmetry microstates of the
	guest molecule.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
result	Output - The symmetry correction.
schema	A serializable schema for this object.
thermodynamic_state	Input - The thermodynamic state that the calculation
	was performed at.

n_microstates

Input - The number of symmetry microstates of the guest molecule. The default value of this attribute is not set and must be set by the user.

Type int

thermodynamic_state

Input - The thermodynamic state that the calculation was performed at. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

result

Output - The symmetry correction. The default value of this attribute is not set and must be set by the user..

Type Observable

allow_merging

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 providied.

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(*file_path*)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

ComputeReferenceWork

class openff.evaluator.protocols.paprika.analysis.**ComputeReferenceWork**(*protocol_id*) Computes the reference state work.

__init__(protocol_id)

Methods

__init__(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
	continues on next page

	ed from previous page
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Table 337 - continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
restraints_path	Input - The file path to the JSON file which contains
	the restraint definitions.
result	Output - The reference state work.
schema	A serializable schema for this object.
thermodynamic_state	Input - The thermodynamic state that the calculation
	was performed at.

thermodynamic_state

Input - The thermodynamic state that the calculation was performed at. The default value of this attribute is not set and must be set by the user.

Type ThermodynamicState

restraints_path

Input - The file path to the JSON file which contains the restraint definitions. This will usually have been generated by a *GenerateXXXRestraints* protocol. The default value of this attribute is not set and must be set by the user..

Type str

result

Output - The reference state work. The default value of this attribute is not set and must be set by the user..

Type Observable

allow_merging

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 providied.

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** (Protocol) 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

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(*file_path*)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

Reweighting

ConcatenateTrajectories	A protocol which concatenates multiple trajectories into a single one.
ConcatenateObservables	A protocol which concatenates multiple ObservableFrame objects into a single ObservableFrame object.
BaseEvaluateEnergies	A base class for protocols which will re-evaluate the en- ergy of a series of configurations for a given set of force field parameters.
BaseMBARProtocol	Re-weights a set of observables using MBAR to calcu- late the average value of the observables at a different state than they were originally measured.
<i>ReweightObservable</i>	Reweight an array of observables to a new state using MBAR.
ReweightDielectricConstant	Computes the avergage value of the dielectric constant be re-weighting a set a set of dipole moments and vol- umes using MBAR.

ConcatenateTrajectories

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
get_class_attribute(reference_path)	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
<pre>input_coordinate_paths</pre>	Input - A list of paths to the starting PDB coordinates
	for each of the trajectories.
<pre>input_trajectory_paths</pre>	Input - A list of paths to the trajectories to concate-
	nate.
output_coordinate_path	Output - The path the PDB coordinate file which
	contains the topology of the concatenated trajectory.
output_trajectory_path	Output - The path to the concatenated trajectory.
outputs	A dictionary of the outputs of this property.
	continues on next page

class openff.evaluator.protocols.reweighting.**ConcatenateTrajectories**(*protocol_id*) A protocol which concatenates multiple trajectories into a single one.

Table 341 – continued from previous page	
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.

input_coordinate_paths

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

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

Output - The path the PDB coordinate file which contains the topology of the concatenated trajectory. The default value of this attribute is not set and must be set by the user.

Type str

output_trajectory_path

Output - The path to the concatenated trajectory. The default value of this attribute is not set and must be set by the user.

Type str

allow_merging

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

ConcatenateObservables

class openff.evaluator.protocols.reweighting.ConcatenateObservables(protocol_id)

A protocol which concatenates multiple ObservableFrame objects into a single ObservableFrame object.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
id	The unique id of this protocol.
input_observables	Input - A list of observable arrays to concatenate.
output_observables	Output - The concatenated observable array.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.

input_observables

Input - A list of observable arrays to concatenate. The default value of this attribute is not set and must be set by the user..

Type list

output_observables

Output - The concatenated observable array. The default value of this attribute is not set and must be set by the user..

Type typing.Union[openff.evaluator.utils.observables.ObservableArray, openff.evaluator.utils.observables.ObservableFrame]

allow_merging

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** (*boo1*) – 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** (Protocol) – 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.
- validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

BaseEvaluateEnergies

class openff.evaluator.protocols.reweighting.BaseEvaluateEnergies(protocol_id)

A base class for protocols which will re-evaluate the energy of a series of configurations for a given set of force field parameters.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
5 5	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
-	takes input from.
enable_pbc	Input - If true, periodic boundary conditions will be
	enabled.
gradient_parameters	Input - An optional list of parameters to differentiate
	the evaluated energies with respect to.
id	The unique id of this protocol.
output_observables	Output - An observable array which stores the re-
	duced potentials potential energies evaluated at the
	specified state and using the specified system object
	for each configuration in the trajectory.
outputs	A dictionary of the outputs of this property.
parameterized_system	Input - The parameterized system object which en-
	codes the systems potential energy function.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
thermodynamic_state	Input - The state to calculate the reduced potentials
	at.

continues on next page

Table 345 – continued from previous page	
<pre>trajectory_file_path</pre>	Input - The path to the trajectory file which contains
	the configurations to calculate the energies of.

thermodynamic_state

Input - The state to calculate the reduced potentials at. The default value of this attribute is not set and must be set by the user.

Type ThermodynamicState

parameterized_system

Input - The parameterized system object which encodes the systems potential energy function. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

enable_pbc

Input - If true, periodic boundary conditions will be enabled. The default value of this attribute is True.

Type bool

trajectory_file_path

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

gradient_parameters

Input - An optional list of parameters to differentiate the evaluated energies with respect to.

Type list

output_observables

Output - An observable array which stores the reduced potentials potential energies evaluated at the specified state and using the specified system object for each configuration in the trajectory. The default value of this attribute is not set and must be set by the user.

Type ObservableFrame

allow_merging

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) - The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

BaseMBARProtocol

class openff.evaluator.protocols.reweighting.BaseMBARProtocol(protocol_id)

Re-weights 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)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
bootstrap_iterations	Input - The number of bootstrap iterations to perform
	if bootstraped uncertainties have been requested The
	default value of this attribute is 250.
bootstrap_uncertainties	Input - If true, bootstrapping will be used to esti-
	mated the total uncertainty in the reweighted value.
	continues on next page

dependencies	A list of pointers to the protocols which this protocol takes input from.
	takes input from.
effective_samples	Output - The number of effective samples which
	were re-weighted.
<pre>frame_counts</pre>	Input - The number of configurations per reference
	state.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
reference_reduced_potentials	Input - The reduced potentials of each configuration
	evaluated at each of the reference states.
required_effective_samples	Input - The minimum number of effective samples
	required to be able to reweight the observable.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
<pre>target_reduced_potentials</pre>	Input - The reduced potentials of each configuration
	evaluated at the target state.
value	Output - The re-weighted average value of the ob-
	servable at the target state.

Table 347 - continued from previous page

reference_reduced_potentials:

List[openff.evaluator.utils.observables.ObservableArray]

Input - The reduced potentials of each configuration evaluated at each of the reference states. The default value of this attribute is not set and must be set by the user.

Type list

target_reduced_potentials

Input - The reduced potentials of each configuration evaluated at the target state. The default value of this attribute is not set and must be set by the user.

Type ObservableArray

frame_counts

Input - The number of configurations per reference state. The sum of theseshould equal the length of the reference_reduced_potentials and target_reduced_potentials input arrays as well any input observable arrays. The default value of this attribute is not set and must be set by the user.

Type list

bootstrap_uncertainties

Input - If true, bootstrapping will be used to estimated the total uncertainty in the reweighted value. The default value of this attribute is False.

Type bool

bootstrap_iterations

Input - The number of bootstrap iterations to perform if bootstraped uncertainties have been requested The default value of this attribute is 250.

Type int

required_effective_samples

Input - The minimum number of effective samples required to be able to reweight the observable. If the effective samples is less than this minimum an exception will be raised. The default value of this attribute is 50.

Type int

value

Output - The re-weighted average value of the observable at the target state. The default value of this attribute is not set and must be set by the user.

Type Observable

effective_samples

Output - The number of effective samples which were re-weighted. The default value of this attribute is not set and must be set by the user.

Type float

allow_merging

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 providied.

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** (Protocol) 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

execute(directory=", available_resources=None)
Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

ReweightObservable

class openff.evaluator.protocols.reweighting.**ReweightObservable**(*protocol id*) Reweight an array of observables to a new state using MBAR.

__init__(protocol_id)

Methods

___init___(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
	continues on port page

continues on next page

	eu nom previous page
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Table 348 - continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
bootstrap_iterations	Input - The number of bootstrap iterations to perform
	if bootstraped uncertainties have been requested The
	default value of this attribute is 250.
bootstrap_uncertainties	Input - If true, bootstrapping will be used to esti-
	mated the total uncertainty in the reweighted value.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
effective_samples	Output - The number of effective samples which
	were re-weighted.
<pre>frame_counts</pre>	Input - The number of configurations per reference
	state.
id	The unique id of this protocol.
observable	Input - The observables to reweight.
outputs	A dictionary of the outputs of this property.
reference_reduced_potentials	Input - The reduced potentials of each configuration
	evaluated at each of the reference states.
required_effective_samples	Input - The minimum number of effective samples
	required to be able to reweight the observable.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
<pre>schema target_reduced_potentials</pre>	Input - The reduced potentials of each configuration
	Input - The reduced potentials of each configuration

observable

Input - The observables to reweight. The array should contain the values of the observable evaluated for of each configuration at the target state. The default value of this attribute is not set and must be set by the

user..

Type ObservableArray

allow_merging

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 providied.

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

Input - The number of bootstrap iterations to perform if bootstraped uncertainties have been requested The default value of this attribute is 250.

Type int

bootstrap_uncertainties

Input - If true, bootstrapping will be used to estimated the total uncertainty in the reweighted value. 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** (Protocol) 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_samples

Output - The number of effective samples which were re-weighted. The default value of this attribute is not set and must be set by the user.

Type float

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

frame_counts

Input - The number of configurations per reference state. The sum of theseshould equal the length of the reference_reduced_potentials and target_reduced_potentials input arrays as well any input observable arrays. The default value of this attribute is not set and must be set by the user.

Type list

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

reference_reduced_potentials:

List[openff.evaluator.utils.observables.ObservableArray]

Input - The reduced potentials of each configuration evaluated at each of the reference states. 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

Input - The minimum number of effective samples required to be able to reweight the observable. If the effective samples is less than this minimum an exception will be raised. The default value of this attribute is 50.

Type int

property required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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

Input - The reduced potentials of each configuration evaluated at the target state. The default value of this attribute is not set and must be set by the user.

Type ObservableArray

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

value

Output - The re-weighted average value of the observable at the target state. The default value of this attribute is not set and must be set by the user.

Type Observable

ReweightDielectricConstant

class openff.evaluator.protocols.reweighting.ReweightDielectricConstant(protocol_id)
 Computes the avergage value of the dielectric constant be re-weighting a set a set of dipole moments and volumes
 using MBAR.

__init__(protocol_id)

Methods

___init___(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
can_merge(other[, path_replacements])	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
bootstrap_iterations	Input - The number of bootstrap iterations to perform
	if bootstraped uncertainties have been requested The
	default value of this attribute is 250.
bootstrap_uncertainties	Input - If true, bootstrapping will be used to esti-
	mated the total uncertainty in the reweighted value.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
dipole_moments	Input - The dipole moments evaluated at reference
	state's configurationsusing the force field of the target
	state.
effective_samples	Output - The number of effective samples which
	were re-weighted.
frame_counts	Input - The number of configurations per reference
	state.
id	The unique id of this protocol.
outputs	A dictionary of the outputs of this property.
reference_reduced_potentials	Input - The reduced potentials of each configuration
	evaluated at each of the reference states.
<pre>required_effective_samples</pre>	Input - The minimum number of effective samples
	required to be able to reweight the observable.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
<pre>target_reduced_potentials</pre>	Input - The reduced potentials of each configuration
	evaluated at the target state.
thermodynamic_state	Input - The thermodynamic state to re-weight to.
value	Output - The re-weighted average value of the ob-
	servable at the target state.
volumes	Input - The dipole moments evaluated at reference
	state's configurationsusing the force field of the target
	state.

dipole_moments

Input - The dipole moments evaluated at reference state's configurations using the force field of the target state. The default value of this attribute is not set and must be set by the user.

Type typing.Union[openff.evaluator.utils.observables.ObservableArray, list]

volumes

Input - The dipole moments evaluated at reference state's configurations using the force field of the target state. The default value of this attribute is not set and must be set by the user.

Type typing.Union[openff.evaluator.utils.observables.ObservableArray, list]

thermodynamic_state

Input - The thermodynamic state to re-weight to. The default value of this attribute is not set and must be set by the user.

Type ThermodynamicState

bootstrap_uncertainties

Input - If true, bootstrapping will be used to estimated the total uncertainty in the reweighted value. The

default value of this attribute is False.

Type bool

allow_merging

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 providied.

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

Input - The number of bootstrap iterations to perform if bootstraped uncertainties have been requested The default value of this attribute is 250.

Type int

can_merge(other, path_replacements=None)

Determines whether this protocol can be merged with another.

Parameters

• **other** (Protocol) – 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_samples

Output - The number of effective samples which were re-weighted. The default value of this attribute is not set and must be set by the user.

Type float

execute(directory=", available_resources=None)
Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

frame_counts

Input - The number of configurations per reference state. The sum of theseshould equal the length of the reference_reduced_potentials and target_reduced_potentials input arrays as well any input observable arrays. The default value of this attribute is not set and must be set by the user.

Type list

```
classmethod from_json(file_path)
```

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

reference_reduced_potentials:

List[openff.evaluator.utils.observables.ObservableArray]

Input - The reduced potentials of each configuration evaluated at each of the reference states. 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

Input - The minimum number of effective samples required to be able to reweight the observable. If the effective samples is less than this minimum an exception will be raised. The default value of this attribute is 50.

Type int

property required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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

Input - The reduced potentials of each configuration evaluated at the target state. The default value of this attribute is not set and must be set by the user.

Type ObservableArray

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to validate.

Raises ValueError or AssertionError -

value

Output - The re-weighted average value of the observable at the target state. The default value of this attribute is not set and must be set by the user.

Type Observable

Simulation

BaseEnergyMinimisation	A base class for protocols which will minimise the po-
	tential energy of a given system.
BaseSimulation	A base class for protocols which will perform a molec-
	ular simulation in a given ensemble and at a specified
	state.

BaseEnergyMinimisation

class openff.evaluator.protocols.simulation.**BaseEnergyMinimisation**(*protocol_id*) A base class for protocols which will minimise the potential energy of a given system.

__init__(protocol_id)

Methods

__init__(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
<i>json</i> ([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
	continues on next page

Table 353 – continued from previous page		
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.	
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.	

- . . ~ - ~ . .

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
dependencies	A list of pointers to the protocols which this protocol
-	takes input from.
enable_pbc	Input - If true, periodic boundary conditions will be
	enabled.
id	The unique id of this protocol.
<pre>input_coordinate_file</pre>	Input - The coordinates to minimise.
max_iterations	Input - The maximum number of iterations to per-
	form.
<pre>output_coordinate_file</pre>	Output - The file path to the minimised coordinates.
outputs	A dictionary of the outputs of this property.
parameterized_system	Input - The parameterized system object which en-
	codes the systems potential energy function.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
tolerance	Input - The energy tolerance to which the system
	should be minimized.

input_coordinate_file

Input - The coordinates to minimise. The default value of this attribute is not set and must be set by the user..

Type str

parameterized_system

Input - The parameterized system object which encodes the systems potential energy function. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

tolerance

Input - The energy tolerance to which the system should be minimized. The default value of this attribute is 10.0 kJ / mol.

Type Quantity

max_iterations

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

Input - If true, periodic boundary conditions will be enabled. The default value of this attribute is True.

Type bool

output_coordinate_file

Output - The file path to the minimised coordinates. The default value of this attribute is not set and must

be set by the user..

Type str

allow_merging

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 providied.

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 (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

```
Return type cls
```

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

BaseSimulation

class openff.evaluator.protocols.simulation.BaseSimulation(protocol_id)

A base class for protocols which will perform a molecular simulation in a given ensemble and at a specified state.

__init__(protocol_id)

Methods

__init__(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_gpu_platforms	Input - If true, the simulation will be performed
	using a GPU if available, otherwise it will be con-
	strained to only using CPUs.
allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
checkpoint_frequency	Input - The frequency (in multiples of out-
	<i>put_frequency</i>) with which to write to a checkpoint
	file, e.g.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
enable_pbc	Input - If true, periodic boundary conditions will be
	enabled.
ensemble	Input - The thermodynamic ensemble to simulate in.
gradient_parameters	Input - An optional list of parameters to differentiate
	the evaluated energies with respect to.
high_precision	Input - If true, the simulation will be run using dou-
	ble precision.
id	The unique id of this protocol.
<pre>input_coordinate_file</pre>	Input - The file path to the starting coordinates.

continues on next page

Table 356 – continued from previous page	
observables	Output - The observables collected during the sim-
	ulation.
output_coordinate_file	Output - The file path to the coordinates of the final
	system configuration.
output_frequency	Input - The frequency (in number of steps) with
	which to write to the output statistics and trajectory
	files.
outputs	A dictionary of the outputs of this property.
parameterized_system	Input - The parameterized system object which en-
	codes the systems potential energy function.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
<pre>steps_per_iteration</pre>	Input - The number of steps to propogate the system
	by at each iteration.
thermodynamic_state	Input - The thermodynamic conditions to simulate
	under The default value of this attribute is not set and
	must be set by the user
thermostat_friction	Input - The thermostat friction coefficient.
timestep	Input - The timestep to evolve the system by at each
	step.
<pre>total_number_of_iterations</pre>	Input - The number of times to propogate the system
	forward by the <i>steps_per_iteration</i> number of steps.
<pre>trajectory_file_path</pre>	Output - The file path to the trajectory sampled dur-
	ing the simulation.

- . .

steps_per_iteration

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

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

Input - The frequency (in number of steps) with which to write to the output statistics and trajectory files. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 3000.

Type int

checkpoint_frequency

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. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 10. This attribute is optional.

Type int

timestep

Input - The timestep to evolve the system by at each step. When two protocols are merged, the largest value

of this attribute from either protocol is retained. The default value of this attribute is 2.0 fs.

Type Quantity

thermodynamic_state

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

Input - The thermodynamic ensemble to simulate in. The default value of this attribute is Ensemble.NPT.

Type Ensemble

thermostat_friction

Input - The thermostat friction coefficient. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 1.0 / ps.

Type Quantity

input_coordinate_file

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

parameterized_system

Input - The parameterized system object which encodes the systems potential energy function. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

enable_pbc

Input - If true, periodic boundary conditions will be enabled. The default value of this attribute is True.

Type bool

allow_gpu_platforms

Input - If true, the simulation will be performed 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

Input - If true, the simulation will be run using double precision. The default value of this attribute is False.

Type bool

gradient_parameters

Input - An optional list of parameters to differentiate the evaluated energies with respect to.

Type list

output_coordinate_file

Output - The file path to the coordinates of the final system configuration. The default value of this attribute is not set and must be set by the user.

Type str

trajectory_file_path

Output - The file path to the trajectory sampled during the simulation. The default value of this attribute is not set and must be set by the user.

Type str

observables

Output - The observables collected during the simulation. The default value of this attribute is not set and must be set by the user.

Type ObservableFrame

allow_merging

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 providied.

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** (Protocol) 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

execute(*directory=''*, *available_resources=None*) Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

Storage

UnpackStoredSimulationData	Loads a StoredSimulationData object from disk, and
	makes its attributes easily accessible to other protocols.

UnpackStoredSimulationData

class openff.evaluator.protocols.storage.**UnpackStoredSimulationData**(*protocol_id*) Loads a *StoredSimulationData* object from disk, and makes its attributes easily accessible to other protocols.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
execute([directory, available_resources])	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
coordinate_file_path	Output - A path to the stored simulation output co-
-	ordinates.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
force_field_path	Output - A path to the force field parameters used to
	generate the stored data.
id	The unique id of this protocol.
observables	Output - The stored observables frame.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
simulation_data_path	Input - A list / tuple which contains both the path to
	the simulation data object, it's ancillary data direc-
	tory, and the force field which was used to generate
	the stored data.
substance	Output - The substance which was stored.
thermodynamic_state	Output - The thermodynamic state which was stored.
	continues on next page

continues on next page

al number of molecules in the stored
to the stored simulation trajectory.

. .

simulation_data_path

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

Output - The substance which was stored. The default value of this attribute is not set and must be set by the user.

Type Substance

total_number_of_molecules

Output - The total number of molecules in the stored system. The default value of this attribute is not set and must be set by the user.

Type int

thermodynamic_state

Output - The thermodynamic state which was stored. The default value of this attribute is not set and must be set by the user.

Type *ThermodynamicState*

observables

Output - The stored observables frame. The default value of this attribute is not set and must be set by the user.

Type ObservableFrame

coordinate_file_path

Output - A path to the stored simulation output coordinates. The default value of this attribute is not set and must be set by the user..

Type str

trajectory_file_path

Output - A path to the stored simulation trajectory. The default value of this attribute is not set and must be set by the user.

Type str

force_field_path

Output - A path to the force field parameters used to generate the stored data. The default value of this attribute is not set and must be set by the user.

Type str

allow_merging

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** (Protocol) 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

Parameters

- directory (str) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (*str*) – The uuid to prepend.

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.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

YANK Free Energies

BaseYankProtocol	An abstract base class for protocols which will performs a set of alchemical free energy simulations using the YANK framework.
LigandReceptorYankProtocol	A protocol for performing ligand-receptor alchemical free energy calculations using the YANK framework.
SolvationYankProtocol	A protocol for estimating the change in free energy upon transferring a solute into a solvent (referred to as solvent 1) from a second solvent (referred to as solvent 2) by performing an alchemical free energy calculation using the YANK framework.

BaseYankProtocol

class openff.evaluator.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.

__init__(protocol_id)

Methods

__init__(protocol_id)

<pre>apply_replicator(replicator, template_values)</pre>	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
get_value(reference_path)	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
validate([attribute_type])	Validate the values of the attributes.

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
checkpoint_interval	Input - The number of iterations between saving
	YANK checkpoint files.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
<pre>free_energy_difference</pre>	Output - The estimated free energy difference be-
	tween the two phases of interest.
gradient_parameters	Input - An optional list of parameters to differentiate
	the estimated free energy with respect to.
id	The unique id of this protocol.
	continues on payt page

continues on next page

Table 662 Continued from previous page	
<pre>number_of_equilibration_iterations</pre>	Input - The number of iterations used for equilibra-
	tion before production run.
number_of_iterations	Input - The number of YANK iterations to perform.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
setup_only	Input - If true, YANK will only create and validate
	the setup files, but not actually run any simulations.
steps_per_iteration	Input - The number of steps per YANK iteration to
	perform.
thermodynamic_state	Input - The state at which to run the calculations.
timestep	Input - The length of the timestep to take.
verbose	Input - Controls whether or not to run YANK at high
	verbosity.

Table 362 - continued from previous page

thermodynamic_state

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

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

Input - The number of YANK iterations to perform. The default value of this attribute is 5000.

Type int

steps_per_iteration

Input - The number of steps per YANK iteration to perform. The default value of this attribute is 500.

Type int

checkpoint_interval

Input - The number of iterations between saving YANK checkpoint files. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 1.

Type int

timestep

Input - The length of the timestep to take. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 2 fs.

Type Quantity

verbose

Input - Controls whether or not to run YANK at high verbosity. The default value of this attribute is False.

Type bool

setup_only

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

gradient_parameters

Input - An optional list of parameters to differentiate the estimated free energy with respect to.

Type list

free_energy_difference

Output - The estimated free energy difference between the two phases of interest. The default value of this attribute is not set and must be set by the user.

Type Observable

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

allow_merging

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** (Protocol) 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

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

```
json(file_path=None, format=False)
```

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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]

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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 openff.evaluator.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

init(protocol_id)	Constructs a new LigandReceptorYankProtocol ob-
	ject.
apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
<pre>can_merge(other[, path_replacements])</pre>	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.

continues on next page

	leu nom previous page
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
<pre>from_schema(schema)</pre>	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
<pre>get_value(reference_path)</pre>	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
<pre>parse_json(string_contents)</pre>	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Table 363 – continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to
	merge with other protocols.
apply_restraints	Input - Determines whether the ligand should be ex-
	plicitly restrained to the receptor in order to stop the
	ligand from temporarily unbinding.
checkpoint_interval	Input - The number of iterations between saving
	YANK checkpoint files.
<pre>complex_electrostatic_lambdas</pre>	Input - The list of electrostatic alchemical states that
	YANK should sample at when calculating the free
	energy of the ligand in complex with the receptor.
complex_steric_lambdas	Input - The list of steric alchemical states that YANK
	should sample at when calculating the free energy of
	the ligand in complex with the receptor.
dependencies	A list of pointers to the protocols which this protocol
	takes input from.
force_field_path	Input - The path to the force field which defines the
	charge method to use for the calculation.
<pre>free_energy_difference</pre>	Output - The estimated free energy difference be-
	tween the two phases of interest.
gradient_parameters	Input - An optional list of parameters to differentiate
	the estimated free energy with respect to.
id	The unique id of this protocol.
<pre>ligand_electrostatic_lambdas</pre>	Input - The list of electrostatic alchemical states that
	YANK should sample at when calculating the free
	energy of the solvated ligand.
ligand_residue_name	Input - The residue name of the ligand.
	continues on next page

	nued from previous page
ligand_steric_lambdas	Input - The list of steric alchemical states that YANK
	should sample at when calculating the free energy of
	the solvated ligand.
<pre>number_of_equilibration_iterations</pre>	Input - The number of iterations used for equilibra-
	tion before production run.
number_of_iterations	Input - The number of YANK iterations to perform.
outputs	A dictionary of the outputs of this property.
<pre>receptor_residue_name</pre>	Input - The residue name of the receptor.
required_inputs	The inputs which must be set on this protocol.
restraint_type	Input - The type of ligand restraint applied, provided
	that apply_restraints is True The default value of this
	attribute is RestraintType.Harmonic.
schema	A serializable schema for this object.
setup_only	Input - If true, YANK will only create and validate
	the setup files, but not actually run any simulations.
<pre>solvated_complex_coordinates</pre>	Input - The file path to the solvated complex coordi-
	nates.
<pre>solvated_complex_system</pre>	Input - The parameterized solvated complex system
	object.
<pre>solvated_complex_trajectory_path</pre>	Output - The file path to the generated ligand trajec-
	tory.
solvated_ligand_coordinates	Input - The file path to the solvated ligand coordi-
	nates.
<pre>solvated_ligand_system</pre>	Input - The parameterized solvated ligand system ob-
	ject.
<pre>solvated_ligand_trajectory_path</pre>	Output - The file path to the generated ligand trajec-
	tory.
steps_per_iteration	Input - The number of steps per YANK iteration to
	perform.
thermodynamic_state	Input - The state at which to run the calculations.
timestep	Input - The length of the timestep to take.
verbose	Input - Controls whether or not to run YANK at high
	verbosity.

Table 364 – continued from previous page

class RestraintType(value)

The types of ligand restraints available within yank.

ligand_residue_name

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

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

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

Input - The parameterized solvated ligand system object. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

solvated_complex_coordinates

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

Input - The parameterized solvated complex system object. The default value of this attribute is not set and must be set by the user.

Type ParameterizedSystem

force_field_path

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

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

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

ligand_electrostatic_lambdas

Input - The list of electrostatic alchemical states that YANK should sample at when calculating the free energy of the solvated ligand. 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 attribute is *optional*.

Type list

ligand_steric_lambdas

Input - The list of steric alchemical states that YANK should sample at when calculating the free energy of the solvated ligand. 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 attribute is *optional*.

Type list

complex_electrostatic_lambdas

Input - The list of electrostatic alchemical states that YANK should sample at when calculating the free energy of the ligand in complex with the receptor. 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 attribute is *optional*.

Type list

complex_steric_lambdas

Input - The list of steric alchemical states that YANK should sample at when calculating the free energy of the ligand in complex with the receptor. 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 attribute is *optional*.

Type list

solvated_ligand_trajectory_path

Output - The file path to the generated ligand trajectory. The default value of this attribute is not set and must be set by the user..

Type str

solvated_complex_trajectory_path

Output - The file path to the generated ligand trajectory. The default value of this attribute is not set and must be set by the user.

Type str

allow_merging

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** (Protocol) 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

Input - The number of iterations between saving YANK checkpoint files. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 1.

Type int

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

free_energy_difference

Output - The estimated free energy difference between the two phases of interest. The default value of this attribute is not set and must be set by the user.

Type Observable

```
classmethod from_json(file_path)
```

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

gradient_parameters

Input - An optional list of parameters to differentiate the estimated free energy with respect to.

Type list

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user..

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- format (bool) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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

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

Input - The number of YANK iterations to perform. The default value of this attribute is 5000.

Type int

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) – The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

```
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

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

Input - The number of steps per YANK iteration to perform. The default value of this attribute is 500.

Type int

thermodynamic_state

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

Input - The length of the timestep to take. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 2 fs.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

verbose

Input - Controls whether or not to run YANK at high verbosity. The default value of this attribute is False.

Type bool

Solvation YankProtocol

class openff.evaluator.protocols.yank.SolvationYankProtocol(protocol_id)

A protocol for estimating the change in free energy upon transferring a solute into a solvent (referred to as solvent 1) from a second solvent (referred to as solvent 2) by performing an alchemical free energy calculation 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)

Methods

___init___(protocol_id)

apply_replicator(replicator, template_values)	Applies a <i>ProtocolReplicator</i> to this protocol.
can_merge(other[, path_replacements])	Determines whether this protocol can be merged with
	another.
<pre>execute([directory, available_resources])</pre>	Execute the protocol.
<pre>from_json(file_path)</pre>	Create this object from a JSON file.
from_schema(schema)	Initializes a protocol from it's schema definition.
<pre>get_attributes([attribute_type])</pre>	Returns all attributes of a specific <i>attribute_type</i> .
	continues on next nage

continues on next page

<pre>get_class_attribute(reference_path)</pre>	Returns one of this protocols, or any of its children's,
	attributes directly (rather than its value).
<pre>get_value(reference_path)</pre>	Returns the value of one of this protocols inputs / out-
	puts.
<pre>get_value_references(input_path)</pre>	Returns a dictionary of references to the protocols
	which one of this protocols inputs (specified by in-
	<i>put_path</i>) takes its value from.
json([file_path, format])	Creates a JSON representation of this class.
merge(other)	Merges another Protocol with this one.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
<pre>replace_protocol(old_id, new_id)</pre>	Finds each input which came from a given protocol
<pre>set_uuid(value)</pre>	Prepend a unique identifier to this protocols id.
<pre>set_value(reference_path, value)</pre>	Sets the value of one of this protocols inputs.
<pre>validate([attribute_type])</pre>	Validate the values of the attributes.

Table 365 – continued from previous page

Attributes

allow_merging	Input - Defines whether this protocols is allowed to merge with other protocols.
checkpoint_interval	Input - The number of iterations between saving YANK checkpoint files.
dependencies	A list of pointers to the protocols which this protocol takes input from.
electrostatic_lambdas_1	Input - The list of electrostatic alchemical states that YANK should sample at.
electrostatic_lambdas_2	Input - The list of electrostatic alchemical states that YANK should sample at.
<pre>free_energy_difference</pre>	Output - The estimated free energy difference be- tween the solute in thefirst solvent and the second sol- vent (i.e.
gradient_parameters	Input - An optional list of parameters to differentiate the estimated free energy with respect to.
id	The unique id of this protocol.
<pre>number_of_equilibration_iterations</pre>	Input - The number of iterations used for equilibra- tion before production run.
number_of_iterations	Input - The number of YANK iterations to perform.
outputs	A dictionary of the outputs of this property.
required_inputs	The inputs which must be set on this protocol.
schema	A serializable schema for this object.
setup_only	Input - If true, YANK will only create and validate the setup files, but not actually run any simulations.
solute	Input - The substance describing the composition of the solute.
<pre>solution_1_coordinates</pre>	Input - The file path to the coordinates of the solute embedded in the first solvent.
<pre>solution_1_free_energy</pre>	Output - The free energy change of transforming the an ideal solute molecule into a fully interacting molecule in the first solvent.
	continues on next page

continues on next page

lable 366 – cor	ntinued from previous page
solution_1_system	Input - The parameterized system object of the solute
	embedded in the first solvent.
<pre>solution_1_trajectory_path</pre>	Output - The file path to the trajectory containing the
	solute in the first solvent.
solution_2_coordinates	Input - The file path to the coordinates of the solute
	embedded in the second solvent.
solution_2_free_energy	Output - The free energy change of transforming
	the an ideal solute molecule into a fully interacting
	molecule in the second solvent.
solution_2_system	Input - The parameterized system object of the solute
	embedded in the second solvent.
solution_2_trajectory_path	Output - The file path to the trajectory containing the
	solute in the second solvent.
solvent_1	Input - The substance describing the composition of
	the first solvent.
solvent_1_coordinate_path	Output - The file path to the coordinates of only the
	first solvent.
solvent_1_trajectory_path	Output - The file path to the trajectory containing
	only the first solvent.
solvent_2	Input - The substance describing the composition of
	the second solvent.
solvent_2_coordinate_path	Output - The file path to the coordinates of only the
	second solvent.
solvent_2_trajectory_path	Output - The file path to the trajectory containing
	only the second solvent.
steps_per_iteration	Input - The number of steps per YANK iteration to
	perform.
<pre>steric_lambdas_1</pre>	Input - The list of steric alchemical states that YANK
	should sample at.
steric_lambdas_2	Input - The list of steric alchemical states that YANK
	should sample at.
thermodynamic_state	Input - The state at which to run the calculations.
timestep	Input - The length of the timestep to take.
verbose	Input - Controls whether or not to run YANK at high
	verbosity.

Table 366 – continued from previous page

solute

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

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

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

solution_1_coordinates

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

solution_1_system

Input - The parameterized 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 ParameterizedSystem

solution_2_coordinates

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

solution_2_system

Input - The parameterized 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 ParameterizedSystem

electrostatic_lambdas_1

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 attribute is *optional*.

Type list

steric_lambdas_1

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 attribute is *optional*.

Type list

electrostatic_lambdas_2

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 attribute is *optional*.

Type list

steric_lambdas_2

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 attribute is *optional*.

Type list

solution_1_free_energy

Output - The free energy change of transforming the an ideal solute molecule into a fully interacting molecule in the first solvent. The default value of this attribute is not set and must be set by the user.

Type Observable

solvent_1_coordinate_path

Output - The file path to the coordinates of only the first solvent. The default value of this attribute is not set and must be set by the user.

Type str

solvent_1_trajectory_path

Output - The file path to the trajectory containing only the first solvent. The default value of this attribute is not set and must be set by the user.

Type str

solution_1_trajectory_path

Output - The file path to the trajectory containing the solute in the first solvent. The default value of this attribute is not set and must be set by the user.

Type str

solution_2_free_energy

Output - The free energy change of transforming the an ideal solute molecule into a fully interacting molecule in the second solvent. The default value of this attribute is not set and must be set by the user.

Type Observable

solvent_2_coordinate_path

Output - The file path to the coordinates of only the second solvent. The default value of this attribute is not set and must be set by the user.

Type str

solvent_2_trajectory_path

Output - The file path to the trajectory containing only the second solvent. The default value of this attribute is not set and must be set by the user.

Type str

solution_2_trajectory_path

Output - The file path to the trajectory containing the solute in the second solvent. The default value of this attribute is not set and must be set by the user.

Type str

free_energy_difference

Output - The estimated free energy difference between the solute in the first solvent and the second solvent (i.e. $G = G_1 - G_2$). The default value of this attribute is not set and must be set by the user.

Type Observable

allow_merging

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** (Protocol) 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

Input - The number of iterations between saving YANK checkpoint files. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 1.

Type int

property dependencies

A list of pointers to the protocols which this protocol takes input from.

Type list of ProtocolPath

execute(directory=", available_resources=None)

Execute the protocol.

Parameters

- **directory** (*str*) The directory to store output data in.
- **available_resources** (ComputeResources) The resources available to execute on. If *None*, the protocol will be executed on a single CPU.

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (str) – The path to load the JSON from.

Returns The parsed class.

Return type cls

classmethod from_schema(schema)

Initializes a protocol from it's schema definition.

Parameters schema (ProtocolSchema) – The schema to initialize the protocol using.

Returns The initialized protocol.

Return type cls

classmethod get_attributes(attribute_type=None)

Returns all attributes of a specific *attribute_type*.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of 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 *in-put_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 (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

gradient_parameters

Input - An optional list of parameters to differentiate the estimated free energy with respect to.

Type list

id

The unique id of this protocol. The default value of this attribute is not set and must be set by the user.

Type str

json(file_path=None, format=False)

Creates a JSON representation of this class.

Parameters

- file_path (str, optional) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

merge(other)

Merges another Protocol with this one. The id of this protocol will remain unchanged.

Parameters other (Protocol) – 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

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

Input - The number of YANK iterations to perform. The default value of this attribute is 5000.

Type int

property outputs

A dictionary of the outputs of this property.

Type dict of ProtocolPath and Any

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (str or bytes) - The typed json string.

Returns The parsed class.

Return type Any

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 required_inputs

The inputs which must be set on this protocol.

Type list of ProtocolPath

property schema

A serializable schema for this object.

Type ProtocolSchema

set_uuid(value)

Prepend a unique identifier to this protocols id. If the id already has a prepended unid, it will be overwritten by this value.

Parameters value (str) – The uuid to prepend.

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

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

Input - The number of steps per YANK iteration to perform. The default value of this attribute is 500.

Type int

thermodynamic_state

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

Input - The length of the timestep to take. When two protocols are merged, the largest value of this attribute from either protocol is retained. The default value of this attribute is 2 fs.

Type Quantity

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

verbose

Input - Controls whether or not to run YANK at high verbosity. The default value of this attribute is False.

Type bool

2.32.11 Workflow Construction Utilities

SimulationProtocols	The common set of protocols which would be required to estimate an observable by running a new molecule sim- ulation.
ReweightingProtocols	The common set of protocols which would be required to re-weight an observable from cached simulation data.
<pre>generate_base_reweighting_protocols</pre>	Constructs a set of protocols which, when combined in a workflow schema, may be executed to reweight a set of cached simulation data to estimate the average value of an observable.
<pre>generate_reweighting_protocols</pre>	
<pre>generate_simulation_protocols</pre>	Constructs a set of protocols which, when combined in a workflow schema, may be executed to run a single sim- ulation to estimate the average value of an observable.

SimulationProtocols

class openff.evaluator.protocols.utils.SimulationProtocols(build_coordinates:

	—
(openff.evaluator.protocols.coordinates.BuildCoordinatesPa
(assign_parameters:
(openff.evaluator.protocols.forcefield.BaseBuildSystem,
	energy_minimisation:
(openff.evaluator.protocols.openmm.OpenMMEnergyMini
	equilibration simulation:
(openff.evaluator.protocols.openmm.OpenMMSimulation,
	production_simulation:
	openff.evaluator.protocols.openmm.OpenMMSimulation,
	analysis_protocol:
	openff.evaluator.protocols.utils.S,
	converge_uncertainty:
	openff.evaluator.workflow.protocols.ProtocolGroup,
	decorrelate_trajectory:
	openff.evaluator.protocols.analysis.DecorrelateTrajectory,
	decorrelate_observables:
	openff.evaluator.protocols.analysis.DecorrelateObservable
	observable by running a new molecule
1	

The common set of protocols which would be required to estimate an observable by running a new molecule simulation.

__init__(*build_coordinates:* openff.evaluator.protocols.coordinates.BuildCoordinatesPackmol, *assign_parameters:* openff.evaluator.protocols.forcefield.BaseBuildSystem, *energy_minimisation:* openff.evaluator.protocols.openmm.OpenMMEnergyMinimisation, *equilibration_simulation:* openff.evaluator.protocols.openmm.OpenMMSimulation, *production_simulation:* openff.evaluator.protocols.openmm.OpenMMSimulation, *analysis_protocol: openff.evaluator.protocols.utils.S, converge_uncertainty:* openff.evaluator.workflow.protocols.ProtocolGroup, *decorrelate_trajectory:* openff.evaluator.protocols.analysis.DecorrelateTrajectory, *decorrelate_observables:* openff.evaluator.protocols.analysis.DecorrelateObservables) → None

Methods

___init___(build_coordinates, ...)

Attributes

build_coordinates

assign_parameters

energy_minimisation

equilibration_simulation

production_simulation

analysis_protocol

converge_uncertainty

decorrelate_trajectory

decorrelate_observables

ReweightingProtocols

class openff.evaluator.protocols.utils.ReweightingProtocols(unpack_stored_data:

openff.evaluator.protocols.storage.UnpackStoredSimulati join_trajectories: openff.evaluator.protocols.reweighting.ConcatenateTraje join_observables: openff.evaluator.protocols.reweighting.ConcatenateObse build_reference_system: openff.evaluator.protocols.forcefield.BaseBuildSystem, evaluate_reference_potential: openff.evaluator.protocols.reweighting.BaseEvaluateEne *build_target_system:* openff.evaluator.protocols.forcefield.BaseBuildSystem, evaluate_target_potential: openff.evaluator.protocols.reweighting.BaseEvaluateEne statistical inefficiency: openff.evaluator.protocols.utils.S, replicate_statistics: openff.evaluator.protocols.miscellaneous.DummyProtocol decorrelate_reference_potential: openff.evaluator.protocols.analysis.DecorrelateObservable decorrelate_target_potential: openff.evaluator.protocols.analysis.DecorrelateObservab decorrelate_observable: openff.evaluator.protocols.analysis.DecorrelateObservable zero_gradients: Optional/openff.evaluator.protocols.gradients.ZeroGradient reweight observable: openff.evaluator.protocols.utils.T)

The common set of protocols which would be required to re-weight an observable from cached simulation data.

Methods

__init__(unpack_stored_data, ...)

Attributes

unpack_stored_data

join_trajectories

join_observables

build_reference_system

evaluate_reference_potential

build_target_system

evaluate_target_potential

statistical_inefficiency

replicate_statistics

decorrelate_reference_potential

decorrelate_target_potential

decorrelate_observable

zero_gradients

reweight_observable

generate_base_reweighting_protocols

openff.evaluator.protocols.utils.generate_base_reweighting_protocols(statistical_inefficiency:

openff.evaluator.protocols.utils.S, reweight_observable: openff.evaluator.protocols.utils.T, replicator_id: str = 'data_replicator', id_suffix: str = '') \rightarrow Tuple[openff.evaluator.protocols.utils.Reweight openff.evaluator.protocols.utils.T], openff.evaluator.workflow.schemas.ProtocolF

Constructs a set of protocols which, when combined in a workflow schema, may be executed to reweight a set of cached simulation data to estimate the average value of an observable.

Parameters

- **statistical_inefficiency** The protocol which will be used to compute the statistical inefficiency and equilibration time of the observable of interest. This information will be used to decorrelate the cached data prior to reweighting.
- **reweight_observable** The MBAR reweighting protocol to use to reweight the observable to the target state. This method will automatically set the reduced potentials on the object.
- replicator_id (str) The id to use for the cached data replicator.
- **id_suffix** (*str*) A string suffix to append to each of the protocol ids.

Returns

- The protocols to add to the workflow, a reference to the average value of the
- estimated observable (an Observable object), and the replicator which will
- clone the workflow for each piece of cached simulation data.

generate_reweighting_protocols

openff.evaluator.protocols.utils.generate_reweighting_protocols(observable_type:

openff.evaluator.utils.observables.ObservableType, replicator_id: str = 'data_replicator', id_suffix: str = '') \rightarrow Tuple[openff.evaluator.protocols.utils.ReweightingPro openff.evaluator.protocols.reweighting.ReweightOb openff.evaluator.workflow.schemas.ProtocolReplicat

generate_simulation_protocols

openff.evaluator.protocols.utils.generate_simulation_protocols(analysis_protocol:

openff.evaluator.protocols.utils.S, $use_target_uncertainty: bool,$ $id_suffix: str = '',$ $conditional_group: Op$ tional[openff.evaluator.protocols.groups.Conditional] $= None, n_molecules: int = 1000)$ \rightarrow Tu ple[openff.evaluator.protocols.utils.SimulationProtocols.utils.ProtocolPath,openff.evaluator.storage.data.StoredSimulationData]

Constructs a set of protocols which, when combined in a workflow schema, may be executed to run a single simulation to estimate the average value of an observable.

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** The protocol which will extract the observable of interest from the generated simulation data.
- **use_target_uncertainty** Whether to run the simulation until the observable is estimated to within the target uncertainty.
- **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.
- **n_molecules** (*int*) The number of molecules to use in the workflow.

Returns

- The protocols to add to the workflow, a reference to the average value of the
- estimated observable (an Observable object), and an object which describes
- the default data from a simulation to store, such as the uncorrelated statistics
- and configurations.

2.32.12 Attribute Utilities

Attribute	A custom descriptor used to add useful metadata to class attributes.
AttributeClass	A base class for objects which require well defined at-
	tributes with additional metadata.
UNDEFINED	A custom type used to differentiate between None val-
	ues, and an undeclared optional value.
PlaceholderValue	A class to act as a place holder for an attribute whose value is not known a priori, but will be set later by some
	specialised code.

Attribute

class openff.evaluator.attributes.**Attribute**(*docstring*, *type_hint*, *de-*

fault_value=<openff.evaluator.attributes.attributes.UndefinedAttribute object>, *optional=False*, *read_only=False*)

A custom descriptor used to add useful metadata to class attributes.

This decorator expects the object to have a matching private field in addition to the public attribute. For example if an object has an attribute *substance*, the object must also have a *_substance* field.

Notes

The attribute class will automatically create this private attribute on the object and populate it with the default value.

__init__(docstring, type_hint, default_value=<openff.evaluator.attributes.attributes.UndefinedAttribute object>, optional=False, read_only=False)

Initializes a new Attribute 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.
- **default_value** (*Any*) The default value for this attribute.
- **optional** (*bool*) Defines whether this is an optional input of a class. If true, the *de*-*fault_value* should be set to *UNDEFINED*.
- **read_only** (*bool*) Defines whether this attribute is read-only.

Methods

__*init__*(docstring, type_hint[, ...]) Initializes a new Attribute object.

AttributeClass

class openff.evaluator.attributes.AttributeClass

A base class for objects which require well defined attributes with additional metadata.

__init__()

Methods

___init__()

from_json(file_path)	Create this object from a JSON file.
get_attributes([attribute_type])	Returns all attributes of a specific <i>attribute_type</i> .
json([file_path, format])	Creates a JSON representation of this class.
parse_json(string_contents)	Parses a typed json string into the corresponding class
	structure.
validate([attribute_type])	Validate the values of the attributes.

validate(attribute_type=None)

Validate the values of the attributes. If *attribute_type* is set, only attributes of that type will be validated.

Parameters attribute_type (type of Attribute, optional) – The type of attribute to validate.

Raises ValueError or AssertionError -

classmethod get_attributes(attribute_type=None)
 Returns all attributes of a specific attribute_type.

Parameters attribute_type (*type of Attribute, optional*) – The type of attribute to search for.

Returns The names of the attributes of the specified type.

Return type list of str

classmethod parse_json(string_contents)

Parses a typed json string into the corresponding class structure.

Parameters string_contents (*str or bytes*) – The typed json string.

Returns The parsed class.

Return type Any

classmethod from_json(file_path)

Create this object from a JSON file.

Parameters file_path (*str*) – The path to load the JSON from.

Returns The parsed class.

Return type cls

json(*file_path=None*, *format=False*) Creates a JSON representation of this class.

Parameters

- **file_path** (*str*, *optional*) The (optional) file path to save the JSON file to.
- **format** (*bool*) Whether to format the JSON or not.

Returns The JSON representation of this class.

Return type str

UNDEFINED

```
openff.evaluator.attributes.UNDEFINED =
```

<openff.evaluator.attributes.attributes.UndefinedAttribute object>

A custom type used to differentiate between None values, and an undeclared optional value.

PlaceholderValue

class openff.evaluator.attributes.PlaceholderValue

A class to act as a place holder for an attribute whose value is not known a priori, but will be set later by some specialised code. This may include the input to a protocol which will be set by a workflow as the output of an executed protocol.

__init__()

Methods

___init__()

2.32.13 Observable Utilities

Observable	A class which stores the mean value of an observable as well as the standard error in the mean.
ObservableArray	A class which stores the value(s) of an observable ob- tained via molecule simulation (or simulation data) as well as optionally the derivatives of the value with re- spect to certain force field parameters.
<i>ObservableType</i>	An enumeration of the common observables which may be extracted from molecular simulations (or simulation data) and stored in an ObservableFrame.
ObservableFrame	A data object for storing and retrieving frames of the thermodynamic observables enumerated by the ObservableType enum.
bootstrap	Bootstrapping a set of observables to compute the aver- age value of the observables as well as the the standard error in the average.

Observable

class openff.evaluator.utils.observables.Observable(value: Op-

```
tional[Union[openff.evaluator.utils.units.Measurement,
    openff.evaluator.utils.units.Quantity]] = None,
    gradients: Op-
tional[List[openff.evaluator.forcefield.gradients.ParameterGradient]
= None)
```

A class which stores the mean value of an observable as well as the standard error in the mean. Optionally, the derivatives of the mean with respect to certain force field parameters may also be stored.

Methods

lear_gradients()	Clears all gradient information.	
Attributes		
rror		
radients		

clear_gradients() Clears all gradient information.

ObservableArray

class openff.evaluator.utils.observables.ObservableArray(value: Op-

tional[openff.evaluator.utils.units.Quantity]
= None, gradients: Optional[List[openff.evaluator.forcefield.gradients.ParameterGn
= None)

A class which stores the value(s) of an observable obtained via molecule simulation (or simulation data) as well as optionally the derivatives of the value with respect to certain force field parameters.

Methods

init([value, gradients])	
<pre>clear_gradients()</pre>	Clears all gradient information.
join(*observables)	Concatenates multiple observables together in the or-
	der that they appear in the args list.
<pre>subset(indices)</pre>	Extracts the subset of the values stored for this ob-
	servable at the specified indices.

Attributes

gradients	
value	The value(s) of the observable.

property value: openff.evaluator.utils.units.Quantity
 The value(s) of the observable.

 $subset(indices: Iterable[int]) \rightarrow openff.evaluator.utils.observables.ObservableArray$

Extracts the subset of the values stored for this observable at the specified indices.

Parameters indices – The indices of the entries to extract.

Returns

Return type The subset of the observable values.

classmethod join(*observables: openff.evaluator.utils.observables.ObservableArray) \rightarrow

 $open {\it ff.evaluator.utils.observables.ObservableArray}$

Concatenates multiple observables together in the order that they appear in the args list.

Parameters observables - The observables to join.

Returns

Return type The concatenated observable object.

clear_gradients()

Clears all gradient information.

ObservableType

class openff.evaluator.utils.observables.ObservableType(value)

An enumeration of the common observables which may be extracted from molecular simulations (or simulation data) and stored in an ObservableFrame.

__init__()

Attributes

PotentialEnergy		
KineticEnergy		
TotalEnergy		
Temperature		
Volume	 	
Density	 	

continues on next page

Table 381 – continued from previous pa	age
--	-----

Enthalpy

ReducedPotential

ObservableFrame

class openff.evaluator.utils.observables. ObservableFrame (<i>observables: Optional[Dict[Union[str</i> ,
openff.evaluator.utils.observables.ObservableType], openff.evaluator.utils.observables.ObservableArray]] = None)
A data object for storing and retrieving frames of the thermodynamic observables enumerated by the ObservableType enum.
<pre>init(observables: Optional[Dict[Union[str, openff.evaluator.utils.observables.ObservableType],</pre>

Methods

init([observables])	
clear()	
<pre>clear_gradients()</pre>	Clears all gradient information for each observable in the frame.
<pre>from_openmm(file_path[, pressure])</pre>	Creates an observable frame from the CSV output of an OpenMM simulation.
get(k[,d])	
items()	
join(*observable_frames)	Joins multiple observable frames together in the order that they appear in the args list.
keys()	
<i>pop</i> (k[,d])	If key is not found, d is returned if given, otherwise KeyError is raised.
<pre>popitem()</pre>	as a 2-tuple; but raise KeyError if D is empty.
<pre>setdefault(k[,d])</pre>	
<pre>subset(indices)</pre>	Extracts the subset of the the array which is located at the specified indices.
update([E,]**F)	If E present and has a .keys() method, does: for k ir E: D[k] = E[k] If E present and lacks .keys() method does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v
values()	

classmethod from_openmm(*file_path: str, pressure: Optional[openff.evaluator.utils.units.Quantity]* = None) $\rightarrow openff.evaluator.utils.observables.ObservableFrame$

Creates an observable frame from the CSV output of an OpenMM simulation.

Parameters

• **file_path** – The file path to the CSV file.

• pressure – The pressure at which the observables in the csv file were collected.

Returns

Return type The imported observables.

subset (*indices: Iterable[int]*) \rightarrow *openff.evaluator.utils.observables.ObservableFrame* Extracts the subset of the the array which is located at the specified indices.

Parameters indices – The indices of the entries to extract.

Returns

Return type The subset of data.

classmethod join(*observable_frames: openff.evaluator.utils.observables.ObservableFrame) \rightarrow openff.evaluator.utils.observables.ObservableFrame

Joins multiple observable frames together in the order that they appear in the args list.

Parameters observable_frames – The observable frames to join.

Returns

Return type The joined observable frame.

clear_gradients()

Clears all gradient information for each observable in the frame.

clear() \rightarrow None. Remove all items from D.

 $get(k[, d]) \rightarrow D[k]$ if k in D, else d. d defaults to None.

items() \rightarrow a set-like object providing a view on D's items

keys() \rightarrow a set-like object providing a view on D's keys

pop $(k[, d]) \rightarrow v$, remove specified key and return the corresponding value. If key is not found, d is returned if given, otherwise KeyError is raised.

popitem() \rightarrow (k, v), remove and return some (key, value) pair as a 2-tuple; but raise KeyError if D is empty.

setdefault(k[, d]) \rightarrow D.get(k,d), also set D[k]=d if k not in D

update([E], **F) \rightarrow None. Update D from mapping/iterable E and F.

If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

values() \rightarrow an object providing a view on D's values

bootstrap

Bootstrapping a set of observables to compute the average value of the observables as well as the standard error in the average.

Parameters

- **bootstrap_function** The function to evaluate at each bootstrap iteration.
- **iterations** The number of bootstrap iterations to perform.
- **relative_sample_size** The percentage sample size to bootstrap over, relative to the size of the full data set.
- **sub_counts** If the data being bootstrapped contains arrays of concatenated sub data (such as when reweighting), this variable can be used to specify the number of items which belong to each subset. Data is then sampled with replacement so that the bootstrap sample contains the correct proportion of data from each subset.

If the data to bootstrap is of the form [x0, x1, x2, y0, y1] for example, then $data_sub_counts=[3, 2]$ and a possible sample may look like [x0, x0, x2, y0, y0], but never [x0, x1, y0, y1, y1].

The sub-counts must sum up to the total length of the data provided to observables.

• **observables** – The observables which will be passed to the bootstrap function. All observables must have the same length.

Returns

Return type The average of the data and the uncertainty in the average.

2.32.14 Plug-in Utilities

Plug-ins

register_default_plugins	Registers the built-in workflow protocols, calculation
	layers and physical properties with the plugin system.
register_external_plugins	Registers any supported plugins found in external pack-
	ages with the plugin system.

register_default_plugins

register_external_plugins

2.33 Release History

Releases 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

2.33.1 0.3.5

Bugfixes

- PR #367: Fix #365 to/from_pandas does not roundtrip.
- PR #368: Fix #364 Parsing an invalid IUPAC name raises an exception rather than a warning.
- PR #371: Fix gradients of non-Quantity parameters.

New Features

- PR #362: Support dask-jobqueue Slurm backend.
- PR #366: Support gradients of handler attributes.

2.33.2 0.3.4

A patch release which adds the option (and enables it by default) to remove working files, such as simulated trajectories, when they are no longer needed.

Behaviour Changes

• PR #349: Working files are deleted by default after an estimation batch completes.

2.33.3 0.3.3

This release facilitates the migration of the *openff-evaluator* package from *omnia* to *conda-forge*. This mainly involves changes which update the package to use the new namespaces introduced in the *openff-tookit* package, rather than the old and now deprecated *openforcefield* namespaces.

Bugfixes

• PR #346: Remove the unsupported *encoding* json kwarg.

New Features

- PR #341: Replace usages of dynamic Pint classes with internal static variants.
- PR #343: Migrate to the new OpenFF Toolkit namespace.
- PR #345: Migrate all reference from *omnia* to *conda-forge*.

2.33.4 0.3.2

This release exposes the option to disable caching of simulation data by an evaluator server. The performance of the local storage backend is currently poor when dealing with large amounts of cached data and hence it may be preferable to disable caching in such cases.

New Features

• PR #337: Expose server option to dis/enable data caching.

2.33.5 0.3.1

This release fixes a bug introduced in version 0.3.0 of this framework, whereby the default workflows for computing excess properties could in rare cases be incorrectly merged leading to downstream protocols taking their inputs from the wrong upstream protocol outputs.

While this bug should not affect most calculations, it is recommended that any production calculations performed using version 0.3.0 of this framework be repeated using version 0.3.1.

Bugfixes

• PR #331: Fixes merging excess properties.

2.33.6 0.3.0

The main feature of this release is the overhauling of how the framework computes the gradients of observables with respect to force field parameters.

In particular, from this release onwards all gradients will be computed using the fluctuation formula (also referred to as the thermodynamic gradient), rather than calculation be the re-weighted finite difference approach (PR #280). In general the two methods produce gradients which are numerically indistinguishable, and so this should not markedly change any scientific output of this framework.

The change was made to, in future, enable better integration with automatic differentiation libraries such as jax, and differentiable simulation engines such as timemachine which readily and rapidly give access to $dU/d\theta_i$.

Additionally, as of version 0.3.0 'known' charges (i.e. those assigned to TIP3P water and ions) are no longer automatically applied when using a SMIRNOFF based force field. This feature was originally included in the framework as the OpenFF toolkit did not support defining charges on specific molecules in the force field itself. This is now fully supported through the LibraryCharges section of a SMIRNOFF force field and hence this workaround is no longer required. From now on all ion and water charges **must** be specified in the SMIRNOFF force field.

Finally, this release includes **beta** support for computing host-guest binding affinities using the attach-pull-release (APR) method through integration with the pAPRika and taproom packages. This support was largely facilitated by the efforts of the paprika authors - David R. Slochower and Jeffry Setiadi.

Bugfixes

- PR #285: Use merged protocols in workflow provenance.
- PR #287: Fix merging of nested protocol inputs

New Features

- PR #262: Initial host-guest binding affinity support via paprika and taproom.
- PR #280: Switch to computing thermodynamic gradients.
- PR #309: Add a date to the timestamp logging output.
- PR #311: Initial solvation free energy gradient support.
- PR #312: Support caching free energy data.
- PR #324: Adds new miscellaneous DummyProtocol protocol.

Behaviour Changes

- PR #280: Migrate to thermodynamic gradients.
- PR #310: The SMIRNOFF protocol no longer applies 'known' charges (i.e. water and ions).
- PR #316: Add library charges to the TIP3P test data file.
- PR #328: Store workflow provenance as serialized string.

- The StatisticsArray array has been completely removed and replaced with a new set of observable (Observable, ObservableArray, ObservableFrame objects (#279, #286).
- The following protocol inputs / outputs have been renamed:
 - SolvationYankProtocol.solvent_X_system -> SolvationYankProtocol.solution_X_system
 - SolvationYankProtocol.solvent_X_coordinates -> SolvationYankProtocol. solution_X_coordinates
 - SolvationYankProtocol.estimated_free_energy -> SolvationYankProtocol. free_energy_difference
- The following classes have been renamed:
 - OpenMMReducedPotentials -> OpenMMEvaluateEnergies.
 - AveragePropertyProtocol -> BaseAverageObservable, ExtractAverageStatistic
 AverageObservable, ExtractUncorrelatedData -> BaseDecorrelateProtocol,
 ExtractUncorrelatedTrajectoryData -> DecorrelateTrajectory,
 ExtractUncorrelatedStatisticsData -> DecorrelateObservables
 - ConcatenateStatistics -> ConcatenateObservables, BaseReducedPotentials -> BaseEvaluateEnergies, ReweightStatistics -> ReweightObservable
- The following classes have been removed:
 - OpenMMGradientPotentials, BaseGradientPotentials, CentralDifferenceGradient
- The final value estimated by a workflow must now be an Observable object which contains any gradient information to return. (#296).

2.33.7 0.2.2

This release adds documentation for how physical properties are computed within the framework (both for this, and for previous releases.

Documentation

• PR #281: Initial pass at physical property documentation.

2.33.8 0.2.1

A patch release offering minor bug fixes and quality of life improvements.

Bugfixes

- PR #259: Adds is_file_and_not_empty and addresses OpenMM failure modes.
- PR #275: Workaround for N substance molecules > user specified maximum.

New Features

- PR #267: Adds workflow protocol to Boltzmann average free energies.
- PR #269: Expose exclude exact amount from max molecule cap.

2.33.9 0.2.0

This release overhauls the frameworks data curation abilities. In particular, it adds

• a significant amount of data filters, including to filter by state, substance composition and chemical functionalities.

and components to

- easily import all of the ThermoML and FreeSolv archives.
- convert between property types (currently density <-> excess molar volume).
- select data points close to a set of target states, and substances which contain specific functionalities (i.e. select only data points measured for ketones, alcohols or alkanes).

More information about the new curation abilities can be found in the documentation here.

New Features

- PR #260: Data set curation overhaul.
- PR #261: Adds PhysicalPropertyDataSet.from_pandas.

Breaking Changes

• All of the PhysicalPropertyDataSet.filter_by_XXX functions have now been removed in favor of the new curation components. See the *documentation* for information about the newly available filters and more.

2.33.10 0.1.2

A patch release offering minor bug fixes and quality of life improvements.

Bugfixes

- PR #254: Fix incompatible protocols being merged due to an id replacement bug.
- PR #255: Fix recursive ThermodynamicState string representation.
- PR #256: Fix incorrect version when installing from tarballs.

2.33.11 0.1.1

A patch release offering minor bug fixes and quality of life improvements.

Bugfixes

- PR #249: Fix replacing protocols of non-existent workflow schema.
- PR #253: Fix antechamber truncating charge file.

Documentation

• PR #252: Use conda-forge for ambertools installation.

2.33.12 0.1.0 - OpenFF Evaluator

Introducing the OpenFF Evaluator! The release marks a significant milestone in the development of this project, and constitutes an almost full redesign of the framework with a focus on stability and ease of use.

Note: because of the extensive changes made throughout the entire framework, this release should almost be considered as an entirely new package. No files produced by previous versions of this will work with this new release.

Clearer Branding

First and foremost, this release marks the complete rebranding from the previously named *propertyestimator* to the new *openff-evaluator* package. This change is accompanied by the introduction of a new **openff** namespace for the package, signifying it's position in the larger Open Force Field infrastructure and piplelines.

What was previously:

```
import propertyestimator
```

now becomes:

```
import openff.evaluator
```

The rebranded package is now shipped on conda under the new name of openff-evaluator:

```
conda install -c conda-forge -c omnia openff-evaluator
```

Markedly Improved Documentation

In addition, the release includes for the first time a significant amount of documentation for using the **`framework and** it's features`_ as well as a collection of user focused tutorials which can be ran directly in the browser.

Support for RDKit

This release almost entirely removes the dependence on OpenEye thanks to support for RDKit almost universally across the framework.

The only remaining instance where OpenEye is still required is for host-guest binding affinity calculations where it is used to perform docking.

Model Validation

Starting with this release almost all models, range from PhysicalProperty entries to ProtocolSchema objects, are now heavily validated to help catch any typos or errors early on.

Batching of Similar Properties

The EvaluatorServer now more intelligently attempts to batch properties which may be computed using the same simulations into a single batch to be estimated. While the behaviour was already supported for pure properties in previous, this has now been significantly expanded to work well with mixture properties.

2.33.13 0.0.9 - Multi-state Reweighting Fix

This release implements a fix for calculating the gradients of properties being estimated by reweighting data cached from multiple independant simulations.

Bugfixes

• PR #143: Fix for multi-state gradient calculations.

2.33.14 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.

• PR #142: All ThermoMLXXX classes other than ThermoMLDataSet are now private.

2.33.15 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.

2.33.16 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.

• 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

2.33.17 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.

2.33.18 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=...)

2.33.19 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 arguemt 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.

- 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

2.33.20 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.

2.33.21 0.0.1 - Initial Release

The initial pre-alpha release of the framework.

2.34 Release Process

This document aims to outline the steps needed to release the openff-evaluator on conda-forge. This should only be done with the approval of the core maintainers.

2.34.1 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
remedied by the follow steps:
* A somewhat verbose guide on how users should upgrade their code given the new breaking.
\hookrightarrow changes.
```

2.34.2 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/openff-evaluator/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.

2.34.3 3: Trigger a New Build on Conda Forge

To trigger the build on conda-forge:

1) Create a fork of the openff-evaluator-feedstock and make the following changes to the recipe/meta.yaml file:

- a) Update the version to match the release.
- b) Set build to 0
- c) Update any dependencies in the requirements section
- d) Update the sha256 hash to the output of curl -sL https://github.com/openforcefield/ openff-evaluator/archive/{{ version }}.tar.gz | openssl sha256
- 2) Open PR to merge the fork into the main feedstock:
- a) The PR title should have the format Release X.Y.Z
- b) No PR body text is needed
- c) The CI will run on this PR (~30 minutes) and attempt to build the package.
- d) If the build is successful the PR should be reviewed and merged by the feedstock maintainers.
- e) **Once merged** the package is built again on and uploaded to anaconda.
- 3) Test the conda-forge package:
- a) conda install -c conda-forge openff-evaluator

2.34.4 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.

BIBLIOGRAPHY

- [1] Alice Glättli, Xavier Daura, and Wilfred F van Gunsteren. Derivation of an improved simple point charge model for liquid water: spc/a and spc/l. *The Journal of chemical physics*, 116(22):9811–9828, 2002.
- [2] Kyle A Beauchamp, Julie M Behr, Ariën S Rustenburg, Christopher I Bayly, Kenneth Kroenlein, and John D Chodera. Toward automated benchmarking of atomistic force fields: neat liquid densities and static dielectric constants from the thermoml data archive. *The Journal of Physical Chemistry B*, 119(40):12912–12920, 2015.
- [3] Junmei Wang and Tingjun Hou. Application of molecular dynamics simulations in molecular property prediction. 1. density and heat of vaporization. *Journal of chemical theory and computation*, 7(7):2151–2165, 2011.
- [4] David R Slochower, Niel M Henriksen, Lee-Ping Wang, John D Chodera, David L Mobley, and Michael K Gilson. Binding thermodynamics of host–guest systems with smirnoff99frosst 1.0. 5 from the open force field initiative. *Journal of Chemical Theory and Computation*, 15(11):6225–6242, 2019.
- [1] John D Chodera. A simple method for automated equilibration detection in molecular simulations. *Journal of chemical theory and computation*, 12(4):1799–1805, 2016.
- [2] Richard A Messerly, S Mostafa Razavi, and Michael R Shirts. Configuration-sampling-based surrogate models for rapid parameterization of non-bonded interactions. *Journal of Chemical Theory and Computation*, 14(6):3144–3162, 2018.
- Lee-Ping Wang, Teresa Head-Gordon, Jay W Ponder, Pengyu Ren, John D Chodera, Peter K Eastman, Todd J Martinez, and Vijay S Pande. Systematic improvement of a classical molecular model of water. *The Journal of Physical Chemistry B*, 117(34):9956–9972, 2013.

INDEX

Symp	OIS	init_	() (openff.evaluator.datasets.PropertyPhase
init_	() (openff.evaluator.attributes.Attribute		method), 96
	method), 564	init_	_() (openff.evaluator.datasets.Source method),
init_	(<i>openff.evaluator.attributes.AttributeClass</i>		97
	method), 564	init_	() (openff.evaluator.datasets.curation.components.CurationCom
init_	() (openff.evaluator.attributes.PlaceholderValue		<i>method</i>), 152
	method), 566	init_	_() (openff.evaluator.datasets.curation.components.CurationCom
init_	() (openff.evaluator.backends.CalculationBacker	nd	<i>method</i>), 153
	method), 203	init_	() (openff.evaluator.datasets.curation.components.conversion.Co
init_	() (openff.evaluator.backends.ComputeResource	s	<i>method</i>), 180
	<i>method</i>), 204	init_	() (openff.evaluator.datasets.curation.components.conversion.Co
init_	() (openff.evaluator.backends.QueueWorkerReso	ources	<i>method</i>), 179
	<i>method</i>), 205	init_	() (openff.evaluator.datasets.curation.components.filtering.Filter
init_	() (openff.evaluator.backends.dask.BaseDaskBa	ckend .	method), 164
	<i>method</i>), 207		() (openff.evaluator.datasets.curation.components.filtering.Filter
init_	() (openff.evaluator.backends.dask.BaseDaskJo	bQueueBa	ickettad), 164
	<i>method</i>), 208	init_	() (openff.evaluator.datasets.curation.components.filtering.Filter
init_	() (openff.evaluator.backends.dask.DaskLSFBac	ckend	method), 161
	<i>method</i>), 211	init_	() (openff.evaluator.datasets.curation.components.filtering.Filter
init_	() (openff.evaluator.backends.dask.DaskLocalC	luster	method), 161
	<i>method</i>), 210		() (openff.evaluator.datasets.curation.components.filtering.Filter
init_	() (openff.evaluator.backends.dask.DaskPBSBac	ckend	method), 171
	<i>method</i>), 213	1n1t_	(openff.evaluator.datasets.curation.components.filtering.Filter
init_		2	method), 171
	method), 80	1n1t_	() (openff.evaluator.datasets.curation.components.filtering.Filter
init_	() (openff.evaluator.client.ConnectionOptions		method), 165
	method), 80	INI(_	() (openff.evaluator.datasets.curation.components.filtering.Filter
init_	1 55	ini+	method), 165 () (openff.evaluator.datasets.curation.components.filtering.Filter
	method), 78	IIII (method), 159
	_() (openff.evaluator.client.Request method), 82	ini+	() (openff.evaluator.datasets.curation.components.filtering.Filter
init_		1111 (method), 159
	method), 84	init	() (openff.evaluator.datasets.curation.components.filtering.Filter
init_			method), 169
	method), 86	init	() (openff.evaluator.datasets.curation.components.filtering.Filter
init_	() (openff.evaluator.datasets.CalculationSource	1111 C_	method), 168
	method), 98	init	
init_	() (openff.evaluator.datasets.MeasurementSourc	e	method), 158
· · · · ·	method), 99	ini†	() (openff.evaluator.datasets.curation.components.filtering.Filter
init_	_() (openff.evaluator.datasets.PhysicalProperty		method), 158
i	method), 93	atainit	
1	_() (openij.evalualor.aatasets.PhysicalPropertyD	utasei	method), 162
	<i>method</i>), 138		,,

 \sim

. .

init_		LitterBentffarventtyffypalsStatsentsacuration.workflow.CurationWorkflow
2	method), 162	method), 154
1n1t_	() (openff.evaluator.datasets.curation.components.filming	
::+	method), 160 _() (openff.evaluator.datasets.curation.component <u>s.filmiring</u>	method), 147
INI(
2	method), 160	method), 151
1n1t_		LittorBotSifietes luator.datasets.thermoml.ThermoMLDataSet
	method), 166	method), 142
init_	() (openff.evaluator.datasets.curation.components.fi lmi tag	
	method), 166	method), 181
init_		.Ki)terBessfitierkeluator.forcefield.LigParGenForceFieldSource
	method), 167	method), 185
init_	() (openff.evaluator.datasets.curation.components.fi lmi tag	
	method), 167	method), 188
init_	() (openff.evaluator.datasets.curation.components.fi lmi ing	
	method), 163	method), 187
init_	() (openff.evaluator.datasets.curation.components.filming	
	method), 163	method), 182
init_	_() (openff.evaluator.datasets.curation.components.filming	
	<i>method</i>), 170	<i>method</i>), 183
init_	_() (openff.evaluator.datasets.curation.component <u>s.filmim</u> g	
	<i>method</i>), 169	<i>method</i>), 188
init_	_() (openff.evaluator.datasets.curation.component <u>s.filmi</u> ng	
	<i>method</i>), 157	<i>method</i>), 189
init_	() (openff.evaluator.datasets.curation.component <u>s.filmim</u> g	KilteoByAffmpahaattoreSahersa CalculationLayerSchema
	<i>method</i>), 157	<i>method</i>), 191
init_	_() (openff.evaluator.datasets.curation.component <u>s.filmin</u> g	.Kilt@panffievadenator.layers.reweighting.ReweightingLayer
	<i>method</i>), 156	<i>method</i>), 199
init_	() (openff.evaluator.datasets.curation.component <u>s.filmiing</u>	Kilt@Panffievada&theha gers.reweighting.ReweightingSchema
	<i>method</i>), 156	<i>method</i>), 201
init_	() (openff.evaluator.datasets.curation.component <u>s.fimialy</u>	LhypopEngESodduator.layers.simulation.SimulationLayer
	<i>method</i>), 173	<i>method</i>), 196
init_	() (openff.evaluator.datasets.curation.components.firmialy	. ImpapEngESodwSahemla yers.simulation.SimulationSchema
	<i>method</i>), 172	<i>method</i>), 197
init_	() (openff.evaluator.datasets.curation.component <u>s.s</u> ähritio	n.FingpeRfi.enTypetor.layers.workflow.WorkflowCalculationLayer
	<i>method</i>), 179	<i>method</i>), 193
init_	() (openff.evaluator.datasets.curation.component <u>s.sabritio</u>	n.Selequenff.@Palutstor.layers.workflow.WorkflowCalculationSchema
	<i>method</i>), 177	<i>method</i>), 194
init_	() (openff.evaluator.datasets.curation.component <u>s.sehritio</u>	<u>n.SelectDataBpintfScohelmator.properties.Density</u>
	<i>method</i>), 176	<i>method</i>), 100
init_	() (openff.evaluator.datasets.curation.component <u>s.sehritio</u>	n.SelectSulfs.stvalcustor.properties.DielectricConstant
	<i>method</i>), 176	<i>method</i>), 107
init_	() (openff.evaluator.datasets.curation.component <u>s.sehritio</u>	n.SelectSulfs.twolcustSolpernperties.EnthalpyOfMixing
	<i>method</i>), 175	method), 111
init_	() (openff.evaluator.datasets.curation.component <u>s.sahai.tio</u>	n.Statepenff.evaluator.properties.EnthalpyOfVaporization
	method), 177	method), 114
init_	() (openff.evaluator.datasets.curation.component <u>s.sethritio</u>	
	method), 178	method), 104
init	() (openff.evaluator.datasets.curation.component <u>s.therirtor</u>	
	<i>method</i>), 174	method), 121
init	() (openff.evaluator.datasets.curation.components.tharing)	
	<i>method</i>), 173	method), 118
init		6. Contemportation of the second state of the
_	<i>method</i>), 153	method), 297

init_	_() (openff.evaluator.protocols.analysis.AverageFreeEnirg method), 303	ies() (openff.evaluator.protocols.paprika.analysis.ComputeReference method), 479
init		<pre>// (openff.evaluator.protocols.paprika.analysis.ComputeSymmetic</pre>
	method), 291	method), 474
init_	() (openff.evaluator.protocols.analysis.BaseAverageildister	valleopenff.evaluator.protocols.paprika.coordinates.AddDummyA
	<i>method</i>), 286	<i>method</i>), 443
init_		<u>olocoppenff.evaluator.protocols.paprika.coordinates.PreparePullC</u>
	method), 313	method), 432
init_		ment@penff.evaluator.protocols.paprika.coordinates.PrepareRelea method), 438
ini+	method), 308	methoa), 438 a&b&openff.evaluator.protocols.paprika.restraints.ApplyRestraints
	method), 324	method), 464
init_		مارض) (openff.evaluator.protocols.paprika.restraints.GenerateAttach
	<i>method</i>), 318	method), 448
init_	() (openff.evaluator.protocols.coordinates.BuildCoordinates	tesPactpeolff.evaluator.protocols.paprika.restraints.GeneratePullRe
	<i>method</i>), 329	<i>method</i>), 453
init_		<u>po</u> (d)hapesnff.evaluator.protocols.paprika.restraints.GenerateReleas
	method), 341	method), 459
init_	_() (openff.evaluator.protocols.coordinates.SolvateExisting method), 335	St(Udprenff.evaluator.protocols.reweighting.BaseEvaluateEnergies method), 494
init_	_() (openff.evaluator.protocols.forcefield.BaseBuil <u>dSistim</u> _	() (openff.evaluator.protocols.reweighting.BaseMBARProtocol
	<i>method</i>), 347	<i>method</i>), 500
init_		yst@mopenff.evaluator.protocols.reweighting.ConcatenateObserval
	method), 357	method), 490
init_		erf() (openff.evaluator.protocols.reweighting.ConcatenateTrajector
ini+	method), 352	method), 485 n_() (openff.evaluator.protocols.reweighting.ReweightDielectricCo
	method), 363	method), 511
init_		() (openff.evaluator.protocols.reweighting.ReweightObservable
	<i>method</i>), 368	method), 505
init_	() (openff.evaluator.protocols.groups.ConditionalGrinipt_	() (openff.evaluator.protocols.simulation.BaseEnergyMinimisation
	<i>method</i>), 374	<i>method</i>), 517
init_	_() (openff.evaluator.protocols.miscellaneous.Add <u>Valurei</u> st_	
	<i>method</i>), 380	<i>method</i>), 522
init_		() (openff.evaluator.protocols.storage.UnpackStoredSimulationD
ini+	method), 394 _() (openff.evaluator.protocols.miscellaneous.Dum <u>m\$Rict(</u>	method), 529
IIII (method), 409	method), 560
init	_() (openff.evaluator.protocols.miscellaneous.Filte <u>rSitbitur</u>	
	<i>method</i>), 404	method), 558
init_	() (openff.evaluator.protocols.miscellaneous.Multiply.Vatu	e() (openff.evaluator.protocols.yank.BaseYankProtocol
	<i>method</i>), 390	method), 536
init_		es() ($openff.evaluator.protocols.yank.LigandReceptorYankProtocols.yankArkArkArkArkArkArkArkArkArkArkArkArkArk$
	<i>method</i>), 385	<i>method</i>), 541
init_		lefy(ctionff.evaluator.protocols.yank.SolvationYankProtocol
	method), 399	method), 549
init_	_() (openff.evaluator.protocols.openmm.OpenMMEnargyM	
ini+	<pre>method), 415init_ _() (openff.evaluator.protocols.openmm.OpenMMEvaluate.</pre>	
1111 U		() (openff.evaluator.storage.LocalFileStorage
init	() (openff.evaluator.protocols.openmm.OpenMMSimulatio	
		() (openff.evaluator.storage.StorageBackend
init_	() (openff.evaluator.protocols.paprika.analysis.AnalyzeAF	
		() (openff.evaluator.storage.attributes.FilePath

	<i>method</i>), 247		<i>method</i>), 266
init_	() (openff.evaluator.storage.attributes.QueryAttr	<i>i<u>bu</u>ie</i> nit_	() (openff.evaluator.workflow.ProtocolGroup
	<i>method</i>), 254		<i>method</i>), 267
init_	() (openff.evaluator.storage.attributes.StorageAtt	<i>r<u>ib</u>ine</i> it_	() (openff.evaluator.workflow.Workflow
	<i>method</i>), 253		<i>method</i>), 255
init_	() (openff.evaluator.storage.data.BaseSimulation	<i>Da</i> ianit_	() (openff.evaluator.workflow.WorkflowGraph
	<i>method</i>), 225		<i>method</i>), 258
init_	() (openff.evaluator.storage.data.BaseStoredDate	ainit_	() (openff.evaluator.workflow.WorkflowResult
	method), 219		<i>method</i>), 259
init_	() (openff.evaluator.storage.data.ForceFieldData	<i></i> init_	() (openff.evaluator.workflow.attributes.BaseMergeBehaviour
	method), 222		<i>method</i>), 281
init_	() (openff.evaluator.storage.data.HashableStored	<i>l<u>D</u>ainit_</i>	() (openff.evaluator.workflow.attributes.InequalityMergeBehavio
	method), 221		method), 281
init_		<i>ta_</i> init_	() (openff.evaluator.workflow.attributes.InputAttribute
	method), 224		method), 282
init		rgv Ðni ðt	() (openff.evaluator.workflow.attributes.MergeBehaviour
	method), 231		method), 281
init		on Dianait	() (openff.evaluator.workflow.attributes.OutputAttribute
	<i>method</i>), 228		method), 283
init		rv init	() (openff.evaluator.workflow.schemas.ProtocolGroupSchema
1.1.1 C_	method), 235	/ <u>J_</u>	method), 274
init		nDinniAu	er(s) (openff.evaluator.workflow.schemas.ProtocolReplicator
	method), 239	112 militigen	method), 276
init		<i>ar</i> vini+	() (openff.evaluator.workflow.schemas.ProtocolSchema
	method), 238	с <u>лу</u> ші с_	method), 272
ini+		ata (i) n i th	_() (openff.evaluator.workflow.schemas.WorkflowSchema
IIII U_	method), 245	u <u>uq</u> nary_	method), 278
init	() (openff.evaluator.storage.query.SimulationDa	ta Oinairt	
	method), 242	ua⊋mar)c_	method), 284
init		ry init	() (openff.evaluator.workflow.utils.ReplicatorValue
	method), 236	/ <u>y_</u> 1111 (method), 283
init_			<i>method</i>), 205
	method), 131	А	
init_			
1111 U	method), 129	absolut	e_tolerance(openff.evaluator.layers.CalculationLayerSchema
ini+			attribute), 191
1111 U_	_() (openff.evaluator.substances.ExactAmount	absolut	e_tolerance(openff.evaluator.layers.reweighting.ReweightingSc
2	method), 132		attribute), 202
1n1t_	_() (openff.evaluator.substances.MoleFraction	absolut	<pre>e_tolerance(openff.evaluator.layers.simulation.SimulationScher</pre>
	method), 134		attribute), 198
init_		absolut	<pre>e_tolerance(openff.evaluator.layers.workflow.WorkflowCalculat</pre>
	method), 125		attribute), 195
init_	() (openff.evaluator.thermodynamics.Thermodyn	antisvate	e_site_location
	<i>method</i>), 136		$(open {\it ff. evaluator. protocols. coordinates. Build Docked Coordinates}$
init_	() (openff.evaluator.utils.observables.Observable	ę	attribute), 342
	<i>method</i>), 566	add_com	<pre>ponent() (openff.evaluator.substances.Substance</pre>
init_	() (openff.evaluator.utils.observables.Observable	eArray	<i>method</i>), 127
	<i>method</i>), 567	add_con	dition() (openff.evaluator.protocols.groups.ConditionalGroup
init_	() (openff.evaluator.utils.observables.Observable	eFrame	<i>method</i>), 376
	<i>method</i>), 569	add_pro	<pre>perties() (openff.evaluator.datasets.PhysicalPropertyDataSet</pre>
init_	() (openff.evaluator.utils.observables.Observable	eType	method), 139
	<i>method</i>), 568	add_pro	<pre>perties() (openff.evaluator.datasets.taproom.TaproomDataSet</pre>
init_			method), 148
	<i>method</i>), 261	add_pro	perties() (openff.evaluator.datasets.thermoml.ThermoMLDataS
init_	() (openff.evaluator.workflow.ProtocolGraph		method), 143

add_protocols() (openff.evaluator.protocols.groups.Cond	
	allow_merging(openff.evaluator.protocols.miscellaneous.AddValues
add_protocols() (openff.evaluator.workflow.ProtocolGrap	
	allow_merging(openff.evaluator.protocols.miscellaneous.DivideValue
add_protocols() (openff.evaluator.workflow.ProtocolGroup	
	allow_merging(openff.evaluator.protocols.miscellaneous.DummyProtoco
add_schema() (<i>openff.evaluator.client.RequestOptions</i>	attribute), 411
	allow_merging(openff.evaluator.protocols.miscellaneous.FilterSubstance
add_workflows() (openff.evaluator.workflow.WorkflowGrd	
method), 258 AddDummyAtoms (class in	allow_merging(openff.evaluator.protocols.miscellaneous.MultiplyValue attribute), 391
	allow_merging (openff.evaluator.protocols.miscellaneous.SubtractValues
443	attribute), 386
	ac)low_merging (openff.evaluator.protocols.miscellaneous.WeightByMoleF
380	attribute), 401
	allow_merging(openff.evaluator.protocols.openmm.OpenMMEnergyMini
(openff.evaluator.protocols.openmm.OpenMMSim	
	allow_merging(openff.evaluator.protocols.openmm.OpenMMEvaluateEn
allow_gpu_platforms	attribute), 428
	aldrow_merging(openff.evaluator.protocols.openmm.OpenMMSimulation
attribute), 525	attribute), 422
allow_merging(openff.evaluator.protocols.analysis.Average)	zlioweauir Gimytantenff.evaluator.protocols.paprika.analysis.AnalyzeAPRI
attribute), 299	attribute), 470
allow_merging(openff.evaluator.protocols.analysis.Average)	<code>æFrowEmengjang</code> (openff.evaluator.protocols.paprika.analysis.ComputeRefe
attribute), 304	attribute), 480
	χeΩlose_melrlging (openff.evaluator.protocols.paprika.analysis.ComputeSym
attribute), 293	attribute), 475
	adrage Other giald (openff.evaluator.protocols.paprika.coordinates.AddDumr
attribute), 288	attribute), 444
	withowsime Pyang Obpenff.evaluator.protocols.paprika.coordinates.PrepareP
attribute), 314	attribute), 434
	ade Dipoluest ging (copenff.evaluator.protocols.paprika.coordinates.PrepareR
attribute), 309	attribute), 439
attribute), 325	edlutaQbaargablg (openff.evaluator.protocols.paprika.restraints.ApplyRestra attribute), 465
	allitute), 405 adliut <u>Tmicging</u> (openff.evaluator.protocols.paprika.restraints.GenerateAtt
attribute), 320	attribute), 449
	allitome, ++> allitomornlangasRy(chrem/f.evaluator.protocols.paprika.restraints.GeneratePu
attribute), 331	attribute), 454
	all Dowked Co indicates of evaluator.protocols.paprika.restraints.GenerateRe
attribute), 343	attribute), 460
allow_merging(openff.evaluator.protocols.coordinates.So	hala and the second state of the second state of the second state and th
attribute), 336	attribute), 496
allow_merging(openff.evaluator.protocols.forcefield.Base.	B៤៤៨ស្វី 🗴 🗛 Blue Contraction (Barbare Contraction Contractic Contr
attribute), 348	attribute), 502
	HigBurmenSystem(openff.evaluator.protocols.reweighting.ConcatenateObset
attribute), 358	attribute), 491
	ងអារចារឲ្យផ្លាំងក្រលួង៣g (openff.evaluator.protocols.reweighting.ConcatenateTraje
attribute), 353	attribute), 486
	HI kow Smateging (openff.evaluator.protocols.reweighting.ReweightDielectri
attribute), 364	attribute), 513
	akidianterging (openff.evaluator.protocols.reweighting.ReweightObserval
attribute), 370	attribute), 507
a110w_metg1ng(openj).evaluator.protocols.groups.Conditi	aldk&rmprging(openff.evaluator.protocols.simulation.BaseEnergyMinimi

attribute), 519	class method), 157
allow_merging(openff.evaluator.protocols.simulation.Basappinlg)	bitom penff. evaluator. datasets. curation. components. filtering. Filter Duplication of the transmission of transmission of the transmission of transm
attribute), 526	class method), 156
allow_merging (openff.evaluator.protocols.storage.Unpackappiegl(\$)	
attribute), 531	class method), 173
allow_merging(openff.evaluator.protocols.yank.BaseYank Rppby 6) attribute), 538	class method), 177
allow_merging(openff.evaluator.protocols.yank.LigandReapprov X) attribute), 545	mlapeotffacelaluator.datasets.curation.components.selection.SelectSu class method), 176
allow_merging(openff.evaluator.protocols.yank.Solvation ¥ppfAFyr()) attribute), 553	d(cpl enff.evaluator.datasets.curation.components.thermoml.Import class method), 174
allow_merging (openff.evaluator.workflow.Protocol at- apply()	(openff.evaluator.datasets.curation.workflow.CurationWorkflow
<pre>tribute), 262 allow_merging(openff.evaluator.workflow.ProtocolGroupapply()</pre>	class method), 153
attribute), 270	<i>method</i>), 235
	(openff.evaluator.storage.query.BaseSimulationDataQuery
amounts (openff.evaluator.substances.Substance at-	method), 241
tribute), 126 apply() AnalyzeAPRPhase (class in) (openff.evaluator.storage.query.ForceFieldQuery method), 238
openff.evaluator.protocols.paprika.analysis), apply()) (openff.evaluator.storage.query.FreeEnergyDataQuery
469 append_uuid() (<i>openff.evaluator.workflow.utils.ProtocolPa</i> thply()	method), 245) (openff.evaluator.storage.query.SimulationDataQuery
<i>method</i>), 285	<i>method</i>), 243
apply() (openff.evaluator.datasets.curation.components.compression) class method), 180	Commenfi Exactust Dervsity Iflata schemas. Protocol Replicator method), 277
apply() (openff.evaluator.datasets.curation.components.CuapplyCa class method), 152	<pre>perploineat() (openff.evaluator.protocols.analysis.AverageDielect method), 299</pre>
apply() (openff.evaluator.datasets.curation.components.filtapinkyFin	
class method), 164	method), 305
apply() (openff.evaluator.datasets.curation.components.filtapinkyFin	
class method), 161	method), 293
apply()(openff.evaluator.datasets.curation.components.filtapplyFit	
class method), 172	<i>method</i>), 288
apply() (openff.evaluator.datasets.curation.components.filtappingyFin	
class method), 165	method), 315
apply() (openff.evaluator.datasets.curation.components.filtapinkyFin class method), 159	method), 310
apply() (openff.evaluator.datasets.curation.components.filtapinkyEin	
class method), 169	method), 325
apply() (openff.evaluator.datasets.curation.components.filtapipkyFin	
class method), 158	method), 320
apply() (openff.evaluator.datasets.curation.components.filtapplyEi	hepByPatpar()Typesnff.evaluator.protocols.coordinates.BuildCoord
class method), 162	<i>method</i>), 331
apply() (openff.evaluator.datasets.curation.components.filtappakyFi	
class method), 160	method), 343
apply() (openff.evaluator.datasets.curation.components.filtapinkyEin class method), 166	RepBySarlars () (openff.evaluator.protocols.coordinates.SolvateExis method), 336
apply() (openff.evaluator.datasets.curation.components.filtapinkyFin	
class method), 168	method), 348
apply() (openff.evaluator.datasets.curation.components.filtappingyEin	h æpByStarteanWehøjsten ff.evaluator.protocols.forcefield.BuildLigParG
class method), 163	<i>method</i>), 358
apply() (openff.evaluator.datasets.curation.components.filtappingyFin	
class method), 170	method), 353
apply() (openff.evaluator.datasets.curation.components.filtappplyFi	т ерву сепренацио penff.evaluator.protocols.forcefield.BuildTLeapSy

method), 364 method), 513	3
apply_replicator() (openff.evaluator.protocols.gradientaffplyGraphicator() (openff.evaluator.protocols.reweighting.ReweightO
method), 370 method), 50°	7
apply_replicator() (openff.evaluator.protocols.groups.CapplitionalGraptor() (openff.evaluator.protocols.simulation.BaseEnergy
method), 376 method), 519	9
apply_replicator() (openff.evaluator.protocols.miscella appalsyAddpaluaa tor() (openff.evaluator.protocols.simulation.BaseSimula
method), 381 method), 520	
apply_replicator() (openff.evaluator.protocols.miscellaappalsyDrepleValueor(
method), 396 method), 53	
apply_replicator() (openff.evaluator.protocols.miscellaapply_nepliteevor@	
method), 411 method), 538	
apply_replicator() (openff.evaluator.protocols.miscellaapply_idepSubstance)	
method), 406 method), 545	
apply_replicator() (openff.evaluator.protocols.miscella appily/hdp)b/datior (
method), 391 method), 553	
apply_replicator() (openff.evaluator.protocols.miscellaappalySureplaidadtoas(
method), 386 method), 264	
apply_replicator() (<i>openff.evaluator.protocols.miscella</i> apply_WeeghtBy&LoveF	
method), 401 method), 27	
apply_replicator()(openff.evaluator.protocols.openmmappdy/MMEtrregy/Misi(misatffævaluator.protocols.yank.LigandReceptorYank
method), 416 attribute), 54	
apply_replicator()(openff.evaluator.protocols.openmm RppdryResEralimte Ene	ergies (class in
method), 428 openff.evalue	ator.protocols.paprika.restraints),
apply_replicator() (openff.evaluator.protocols.openmm.OpenMMStimulation	
<i>method</i>), 422 assigned_residue_:	names
apply_replicator() (openff.evaluator.protocols.paprika.analysis.AnademfAAVRI	
method), 471 attribute), 33	-
apply_replicator() (openff.evaluator.protocols.paprika. asslygixed ompsicRuge	
	uator.protocols.coordinates.SolvateExistingStructure
apply_replicator() (openff.evaluator.protocols.paprika.analysis.CathributeS)	
	enff.evaluator.protocols.paprika.restraints.GenerateA
apply_replicator() (openff.evaluator.protocols.paprika.coordinatestAidadDa)n##	
	enff.evaluator.protocols.paprika.restraints.Generatel
apply_replicator() (openff.evaluator.protocols.paprika.coordinatestPileupter)e Rt	
	ppenff.evaluator.attributes), 564
apply_replicator()(<i>openff.evaluator.protocols.paprika</i> . AnamibuteePraps r (R	elexasieCopedffiateduator.attributes),
<i>method</i>), 439 564	
apply_replicator() (openff.evaluator.protocols.paprika. AccentigesDApplyRerira	Gon stant (class in
method), 465 openff.evalue	ator.protocols.analysis), 297
apply_replicator() (openff.evaluator.protocols.paprika. AesenaigesFGenEmarshi a	es hRestraints (class in
method), 449 openff.evalue	ator.protocols.analysis), 303
apply_replicator() (openff.evaluator.protocols.paprika.kesemaigeObservable	
	ator.protocols.analysis), 291
apply_replicator() (openff.evaluator.protocols.paprika.restraints.GenerateRel	
method), 460	
apply_replicator() (openff.evaluator.protocols.reweight	
method), 496 openff.evalue	ator.protocols.analysis), 286
apply_replicator() (openff.evaluator.protocols.reweighting Brog Mark Remoc	×
method), 502 openff.evalue	ator.protocols.forcefield), 347
apply_replicator() (openff.evaluator.protocols.reweighting Epysteric Alser	rvables (class in
method), 491 openff.evalue	ator.backends.dask), 207
apply_replicator() (openff.evaluator.protocols.reweightings Constitution ()	ACKEnd (class in
method), 486 openff.evalue	ator.backends.dask), 208
apply_replicator() (openff.evaluator.protocols.reweighting.ReweightDielectric	<i>cConstant</i>

BaseDataQuery (class i	in	(openff.evaluator.protocols.analysis.AverageObservable
openff.evaluator.storage.query), 235		attribute), 294
BaseDecorrelateProtocol (class i	<i>in</i> bootst	rap_sample_size
openff.evaluator.protocols.analysis), 313		$(open {\it ff.evaluator.protocols.analysis.} Base Average Observable$
BaseEnergyMinimisation (class i	in	attribute), 287
openff.evaluator.protocols.simulation), 517	bootst	rap_uncertainties
BaseEvaluateEnergies (class i	in	$(open {\it ff.evaluator.protocols.reweighting.} Base {\it MBARProtocol}$
openff.evaluator.protocols.reweighting), 494		attribute), 501
	<i>in</i> bootst	rap_uncertainties
openff.evaluator.protocols.reweighting), 500		(openff.evaluator.protocols.reweighting.ReweightDielectricConstant (openff.evaluator.protocols.reweight)
5	in	attribute), 512
openff.evaluator.workflow.attributes), 281	bootst	rap_uncertainties
·	in	$(open {\it ff.evaluator.protocols.reweighting.} Reweight Observable$
openff.evaluator.protocols.simulation), 522		attribute), 507
× ×	<i>in</i> box_as	$\verb+pect_ratio(openff.evaluator.protocols.coordinates.BuildCoordinates)$
openff.evaluator.storage.data), 225		attribute), 330
	<i>in</i> box_as	$\verb"pect_ratio" (openff.evaluator.protocols.coordinates.SolvateExisting test in the second state of the se$
openff.evaluator.storage.query), 239		attribute), 337
BaseStoredData (class i	in BuildC	CoordinatesPackmol (class in
openff.evaluator.storage.data), 219		openff.evaluator.protocols.coordinates), 329
BaseYankProtocol (class i	<i>in</i> BuildD	ockedCoordinates (class in
openff.evaluator.protocols.yank), 536		openff.evaluator.protocols.coordinates), 341
Batch (class in openff.evaluator.server), 90	BuildD	ockedCoordinates.ActivateSiteLocation
<pre>batch_mode (openff.evaluator.client.RequestOptions a)</pre>	t-	(class in openff.evaluator.protocols.coordinates),
tribute), 84		342
BatchMode (class in openff.evaluator.client), 80	BuildL	.igParGenSystem (class in
beta (openff.evaluator.thermodynamics.Thermodynamic	cState	openff.evaluator.protocols.forcefield), 357
property), 137	BuildL	.igParGenSystem.WaterModel (class in
bootstrap() (in modul	le	openff.evaluator.protocols.forcefield), 358
openff.evaluator.utils.observables), 571	BuildS	mirnoffSystem (class in
bootstrap_cycles(openff.evaluator.protocols.analysi	is.AverageFr	ree &panffies aluator.protocols.forcefield), 352
attribute), 304	BuildT	LeapSystem (class in
bootstrap_iterations		openff.evaluator.protocols.forcefield), 363
(openff.evaluator.protocols.analysis.AverageD	iele Brai c Cd F	
attribute), 299		openff.evaluator.protocols.forcefield), 364
bootstrap_iterations	BuildT	LeapSystem.WaterModel (class in
(openff.evaluator.protocols.analysis.AverageO	bservable	openff.evaluator.protocols.forcefield), 364
attribute), 294	_	
bootstrap_iterations	С	
(openff.evaluator.protocols.analysis.BaseAvera	ageObserva	ble aqueous ionic mole fraction()
attribute), 287	° cuicui	(openff.evaluator.substances.Substance static
bootstrap_iterations		method), 128
(openff.evaluator.protocols.reweighting.BaseM	1BARProtox	(in module)
attribute), 501	cuicui	openff.evaluator.layers), 192
bootstrap_iterations	calcul	ation_layers (openff.evaluator.client.RequestOptions
(openff.evaluator.protocols.reweighting.Rewei	ghtDielectri	ic Canstant 84
attribute), 513		ation_schemas
bootstrap_iterations	curcur	(openff.evaluator.client.RequestOptions at-
(openff.evaluator.protocols.reweighting.Rewei	ghtObserva	bleribute). 84
attribute), 507		ationBackend (class in
<pre>bootstrap_sample_size</pre>		openff.evaluator.backends), 203
	ielectricGov	astaronLayer (class in openff.evaluator.layers),
attribute), 299	carcar	188
<pre>bootstrap_sample_size</pre>		

OpenFF Evaluator Documentation

CalculationLaye openff.evd	rResult <i>luator.layers</i>),	(<i>class</i> 189	in	can_mei	cge() (openff.evaluator.protocols.openmm.OpenMMEnergyMinimi method), 416
CalculationLaye	•	(class	in	can_mei	<pre>cge() (openff.evaluator.protocols.openmm.OpenMMEvaluateEnerg method), 428</pre>
CalculationSour		(class	in	can_me	cge() (openff.evaluator.protocols.openmm.OpenMMSimulation method), 422
1 00	ff.evaluator.pro		erage	e Driache onveir	<pre>method), 422 Got()t(npenff.evaluator.protocols.paprika.analysis.AnalyzeAPRPha method), 471</pre>
<pre>can_merge() (open</pre>	nff.evaluator.pro	otocols.analysis.Av	erage	eFanEmen	gje() (openff.evaluator.protocols.paprika.analysis.ComputeReferer
-	nff.evaluator.pro	otocols.analysis.Av	erage	e Odane <u>r</u> mel	method), 481 htge() (openff.evaluator.protocols.paprika.analysis.ComputeSymme
method),		otocols analysis Re	150 11		method), 476 g gaDl (openff.evaluator.protocols.paprika.coordinates.AddDummy/
method),	289			-	<i>method</i>), 445
method),	315				PgatOctopenff.evaluator.protocols.paprika.coordinates.PreparePull method), 434
<pre>can_merge() (open method),</pre>		otocols.analysis.Co	отри	te Dap<u>o</u>tud a	fgme@utsopenff.evaluator.protocols.paprika.coordinates.PrepareRele method), 439
	nff.evaluator.pro	otocols.analysis.De	ecorre	el aærQlmæ r	rgakle&openff.evaluator.protocols.paprika.restraints.ApplyRestraint method), 466
<pre>can_merge() (open method),</pre>		otocols.analysis.De	ecorre	el aterT<u>m</u>je r	cge(y) (openff.evaluator.protocols.paprika.restraints.GenerateAttaci method), 450
	nff.evaluator.pro	otocols.coordinates	s.Buil	dCano <u>r</u> dian	<pre>method), 450 gesPa(appeolff.evaluator.protocols.paprika.restraints.GeneratePullR method), 455</pre>
<pre>can_merge() (open</pre>	nff.evaluator.pro	otocols.coordinates	s.Buil	d Danck med	method), 455 Ege(dikopesnff.evaluator.protocols.paprika.restraints.GenerateRelea method), 461
-	nff.evaluator.pro	otocols.coordinates	s.Solv	a caEn<u>xi</u>nne n	gsated droppenff.evaluator.protocols.reweighting.BaseEvaluateEnergia
method),		otocols forcefield B	DasaR	uddhuman	<pre>method), 497 mge() (openff.evaluator.protocols.reweighting.BaseMBARProtocol</pre>
method),	349				<i>method</i>), 502
method),	359			-	Spst@n (openff.evaluator.protocols.reweighting.ConcatenateObserva method), 491
<pre>can_merge() (open method),</pre>		otocols.forcefield.B	BuildS	n cian<u>o</u>ffey	xger(f) (openff.evaluator.protocols.reweighting.ConcatenateTrajecto method), 487
<pre>can_merge() (open method),</pre>		otocols.forcefield.B	BuildT	<u>Tæðup S</u> men	2ge() (openff.evaluator.protocols.reweighting.ReweightDielectricC method), 513
<pre>can_merge() (open method),</pre>		otocols.gradients.Z	leroG	<i>radia</i> nne:	cge() (openff.evaluator.protocols.reweighting.ReweightObservable method), 507
	ff.evaluator.pro	otocols.groups.Cor	nditio	n akin<u>r</u>me p	<pre>cge() (openff.evaluator.protocols.simulation.BaseEnergyMinimisat method), 519</pre>
	nff.evaluator.pro	otocols.miscellanee	ous.A	d d an lune au	rge() (openff.evaluator.protocols.simulation.BaseSimulation method), 526
	ff.evaluator.pro	otocols.miscellane	ous.D)ivialn <u>V</u> mleu	<pre>ge() (openff.evaluator.protocols.storage.UnpackStoredSimulation) method), 532</pre>
	nff.evaluator.pro	otocols.miscellane	ous.D	uoany flex	tge() (openff.evaluator.protocols.yank.BaseYankProtocol method), 539
	nff.evaluator.pro	otocols.miscellane	ous.F	iltanS <u>u</u> hen	method), 555 mge(B)(Rplanff.evaluator.protocols.yank.LigandReceptorYankProtoco method), 545
	ff.evaluator.pro	otocols.miscellane	ous.M	ludaiplynken	<pre>https://www.sets.com/ank/protocols.yank.SolvationYankProtocol method), 554</pre>
<pre>can_merge() (oper</pre>		otocols.miscellaned	ous.S	u lotara <u>c</u> tiVen	
method),					method), 263
<pre>can_merge() (open method),</pre>		otocols.miscellaned	ous. W	/ecghnBnyA	wef) a(ajumff.evaluator.workflow.ProtocolGroup method), 269

method), 249 components (openff evaluator.substances.Substance at- induct), 249 components_only (openff.evaluator.storage.query.SubstanceQuery attribute), 237 (openff.evaluator.storage.query.SubstanceQuery attribute), 336 openff.evaluator.protocols.gorefield.buildEtypeRefere enceGork (class in attribute), 354 openff.evaluator.protocols.gorefield.buildEtypeRefere enceGork (class in (openff.evaluator.protocols.gorefield.buildEtypeRefere enceGork), 204 checkpoint_frequency (appl.evaluator.protocols.gorefield.buildEtypeRefere enceGork), 204 checkpoint_frequency (class in (openff.evaluator.protocols.gorefield.buildEtypeRefere enceGork), 204 checkpoint_interval openff.evaluator.protocols.gorefield.buildEtypeRefere enceGork), 204 attribute), 537 ConcatenateObservables (class in checkpoint_interval openff.evaluator.protocols.gorefield.buildEtypeReferenceGork), 204 attribute), 537 ConcatenateObservables (class in checkpoint_interval openff.evaluator.protocols.gorekisting), 490 (openff.evaluator.protocols.yank.LigandReceptorYliokEttemakeTrajectories (class in (openff.evaluator.protocols.yank.LigandReceptorYliokEttemakeTrajectories (class in (openff.evaluator.protocols.yank.SolvationTakritocol openff.evaluator.protocols.group), 375 checkpoint_interval conditionalGroup (class in (openff.evaluator.protocols.yank.SolvationTakritocol openff.evaluator.protocols.group), 375 method), 500 ConditionalGroup.ConditionalGroup (class in (openff.evaluator.protocols.groups.SolverableEtyperf.evaluator.protocols.groups), 375 method), 500 ConditionalGroup.ConditionalGroup class in class in conff.evaluator.protocols.groups), 375 method), 500 ConditionalGroup.ConditionalGroup (openff.evaluator.protocols.paprika.restraints.Geneerate/Restarbibetist)) (op	<pre>capitalize() (openff.evaluator.storage.attributes.FilePat</pre>	th	129		
method), 249 components (openff evaluator.substances.Substance at- induct) (openff.evaluator.storage.attributes.FilePath method), 249 components_only (openff.evaluator.storage.query.SubstanceQuery attribute), 336 constructions.Solvaties.SolvateEx.Solvations/SolvateExCompStoreDeff.evaluator.storage.query.SubstanceQuery attribute), 336 openff.evaluator.protocols.gorefield.BuildEtypeRSeter enceCoNr (class in attribute), 336 openff.evaluator.protocols.gorefield.BuildEtypeRSeter enceCoNr (class in attribute), 354 openff.evaluator.protocols.gorefield.BuildEtypeRSeter enceCoNr (class in (openff.evaluator.protocols.gorefield.BuildEtypeRSeter enceCoNr (class in (openff.evaluator.protocols.gorefield.BuildEtypeRSeter enceCoNr (class in (openff.evaluator.protocols.gorefield.BuildEtypeRSeter enceCoNr (class in (openff.evaluator.protocols.gorefield.BuildEtypeRSeter enceCoNr (class in (openff.evaluator.protocols.gorefield.BuildEtypeRseter.scienceS.goUToolki (class in (openff.evaluator.protocols.gorefield.BuildEtypeRseter.scienceS.goUToolki (class in (openff.evaluator.protocols.gorefield.BuildEtypeRseter.scienceS.goVIoolki (class in (openff.evaluator.protocols.gorefield.BuildEtypeRseter.scienceS.goVIoolki (class in (openff.evaluator.protocols.gorefield.BuildEtypeRseter.scienceS.goVIoolki (class in checkpoint_interval openff.evaluator.protocols.gore/scienceS.goVIoolki (class in checkpoint_interval openff.evaluator.protocols.gove), 374 attribute), 554 conditionalGroup (class in (openff.evaluator.protocols.gov.scienceS.goVI.goVI) (openff.evaluator.protocols.gove.scienceS.goVI.goVI.goVI.goVI.goVI.goVI.goVI.goVI	<i>method</i>), 249	componen	t_roles(<i>openff.evalue</i>	utor.protocols.misce	llaneous.FilterSubstan
center () (openflevaluator.storage.attributes.FilePah tribuic), 126 components.only (openflevaluator.storage.query.SubstanceQuery attribute), 237 (openflevaluator.protocols.coordinates.SolvateExGangSteeDupelExampleTextual contexts.only (openflevaluator.protocols.coordinates.SolvateExGangSteeDupelExampleTextual contexts.only (openflevaluator.protocols.porefield.buildTigperSectarenceWork (class in attribute), 304 attribute), 304 attribute), 304 attribute, 304 arttribute), 304 (class in openflevaluator.protocols.paprika.analysis), 308 (class in attribute), 304 attribute), 304 arttribute), 304 arttribute), 304 attribute, 304 attribute), 504 (const attribute), 504 (class in openflevaluator.protocols.paprika.analysis), 417 (openflevaluator.protocols.yank.BaseYankProtocol 474 attribute), 537 (concententeObservables (class in openflevaluator.protocols.paprika.analysis), 410 (openflevaluator.protocols.yank.BaseYankProtocol 474 attribute), 536 (const attribute), 536 (const attribute), 546 (const attribute), 547 (const attribute), 546 (const attribute), 546 (const attribute), 546 (const attribute), 547 (const attribute), 548 (const attribute), 548 (const attribute), 540 (const attri	<pre>casefold() (openff.evaluator.storage.attributes.FilePath</pre>		,,		
method), 249 components_only (openff.evaluator.storage.query.SubstanceQuery attribute), 237 center_solute_in_box attribute), 237 attribute), 336 attribute), 336 charge_backend (openff.evaluator.protocols.forcefield.buildThypeReferenceColork (class attribute), 336 in attribute), 336 openff.evaluator.protocols.paprika.analysis), 308 checkpoint_frequency 479 (openff.evaluator.protocols.simulation.BaseSimulation openff.evaluator.backeds), 204 checkpoint_frequency ComputeResources.GPUTOolkit (class (openff.evaluator.protocols.simulation.BaseSimulation openff.evaluator.backeds), 204 checkpoint_interval openff.evaluator.protocols.paprika.analysis), (openff.evaluator.protocols.yank.BaseYankProtocol 474 attribute), 537 ConcattenateObservables (class (openff.evaluator.protocols.yank.LigandReceptorYEndeterrates (class in (openff.evaluator.protocols.yank.SolvationYAnkProtocol openff.evaluator.protocols.group), 374 attribute), 537 ConditionalGroup (class in (openff.evaluator.protocols.group), 374 attribute), 549 ConditionalGroup. Conditional (class				ubstances.Substance	at-
center_solute_in_box attribute, 237 (openff.evaluator.protocols.coordinates.Solvate.ExiGongStevBipel@Homents (class in attribute), 336 charge_backend (openff.evaluator.protocols.coordinates.Solvate.ExiGongStevBipel@Homents (class in attribute), 364 checkpoint_frequency 479 (openff.evaluator.protocols.opennm.OpenMMSimiliariuteResources (class in attribute), 422 checkpoint_frequency ComputeResources.OPUTOOLkit (class in openff.evaluator.protocols.paprika.analysis), checkpoint_frequency ComputeResources.OPUTOOLkit (class in openff.evaluator.protocols.paprika.analysis), (openff.evaluator.protocols.simulation.BaseSimulation openff.evaluator.backends), 205 attribute), 524 ConcatenateObservables (class in openff.evaluator.protocols.paprika.analysis), (openff.evaluator.protocols.yank.BaseYankProtocol 474 attribute), 537 ConcatenateObservables (class in openff.evaluator.protocols.paprika.analysis), (openff.evaluator.protocols.yank.BaseYankProtocol apenff.evaluator.protocols.reveighting), 490 (openff.evaluator.protocols.yank.BaseYankProtocol openff.evaluator.protocols.reveighting), 485 checkpoint_interval openff.evaluator.protocols.groups), 374 attribute), 534 ConditionalGroup (class in openff.evaluator.protocols.groups), 375 method), 570 ConditionalGroup (class in class in class), 554 ConditionalGroup.Condition (class in clear openff.evaluator.protocols.groups), 375 method), 567 ConditionalGroup.Condition Trype (class in clear openff.evaluator.protocols.groups), 375 method), 568 ConditionalGroup.Gondition.Type (class in clear openff.evaluator.protocols.groups), 375 method), 569 ConditionalGroup.Gondition.Type (class in clear openff.evaluator.protocols.groups), 375 method), 568 ConditionalGroup.Gondition.Type (class in clear openff.evaluator.protocols.groups), 375 method), 569 ConditionalGroup.Gondition.Type (class in clear openff.evaluator.protocols.groups), 375 method), 569 ConditionalGroup.Gondition.Type (class in clear openff.evaluator.protocols.groups), 375 method), 569 ConditionalGroup.Gondition.Type (class in clea			<i>,,</i>	~	
(openff evaluator, protocols, coordinates, Solvate ExGong Strebignel & Konnents (class in attribute), 336 charge_backend (openff, evaluator, protocols, force field, BuilGiffiger/Skefare ence Work (class in attribute), 564 openff, evaluator, protocols, opennum. Open/MMSimflomignet Resources (class in attribute), 427 (openff, evaluator, protocols, opennum. Open/MMSimflomignet Resources (class in (openff, evaluator, protocols, opennum. Open/MMSimflomignet Resources (class in (openff, evaluator, protocols, simulation apenff, evaluator, backends), 204 checkpoint_frequency ComputeResources. GPUToolkit (class in (openff, evaluator, protocols, simulation, BaseSimulation openff, evaluator, backends), 205 attribute), 532 (openff, evaluator, protocols, simulation, BaseSimulation openff, evaluator, protocols, paperka, analysis), (openff, evaluator, protocols, symk, BaseYank Protocol 474 attribute), 537 ConcatenateObservables (class in checkpoint_interval openff, evaluator, protocols, reweighting), 490 (openff, evaluator, protocols, symk, Ligand Receptor Filosk Paronak Protocols, reweighting), 490 (openff, evaluator, protocols, symk, Ligand Receptor Filosk Paronak Protocols, reweighting), 455 checkpoint_interval Conditional Group (class in durbate), 554 (class in clear C) (openff, evaluator, protocols, symk, Solvation Yank Protocol openff, evaluator, protocols, groups), 374 attribute), 555 (class in clear_C) (openff, evaluator, utils, observables. Observables Prane openff, evaluator, protocols, groups), 375 method), 570 Conditional Group Condition (class in clear_C) (openff, evaluator, utils, observables. Observable Resource, synthese, synthese (openff, evaluator, utils, observables. Observables, Prane openff, evaluator, protocols, groups), 375 method), 570 Conditional Group. Condition (class in durbate), 550 confidence_intervals clear_condinate_Openff, evaluator, utils, observables. Observables Report, 376 method), 570 (openff, evaluator, protocols, paprika, restraints. Genetarter Attabuksistry), 375 method), 570 (openff,				utor.storage.query.Si	ubstanceQuery
attribute), 336 openfl.evaluator.protocols.analysis), charge_backend (openfl.evaluator.protocols.forcefield.BuilCHippoSpeferenceCeNork (class in attribute), 364 openfl.evaluator.protocols.paprika.analysis), theckpoint_frequency CompetResources (class in (openfl.evaluator.protocols.openum.OpenMMSimiliationt Ressources (class in (openfl.evaluator.protocols.simulation.BaseSimulation openfl.evaluator.backends), 204 (class in (checkpoint_interguency ComputeResources.GPUTOOLKIt (class in (openfl.evaluator.protocols.simulation.BaseSimulation openfl.evaluator.protocols.paprika.analysis), (dopenfl.evaluator.protocols.yank.BaseYankProtocol 474 attribute), 537 ConcatenateObservables (class in (openfl.evaluator.protocols.yank.LigandReceptorViandBatenateTrijectoris (class in (attribute), 554 ConditionalGroup (class in (openfl.evaluator.protocols.yank.SolvationYankProtocol openfl.evaluator.protocols.groups), 375 method), 560 ConditionalGroup.Condition (class in (clear (openfl.evaluator.utils.observables.Observables/Ampfl.evaluator.protocols.groups), 375 in (openfl.evaluator.utils.observables.Observables/Ampfl.evaluator.protocols.groups), 375 in (openfl.evaluator.utils.observables.Observab			,.	<i>(</i> 1	
charge_backend (openff_evaluator.protocols.forcefield.BuilDiffunctSperianenceWork (class in attribute), 364 openff_evaluator.protocols.paprika.analysis), 479 (openff_evaluator.protocols.opennm.OpenhMSimUniteResources (class in attribute), 422 openff_evaluator.backends), 204 (class in (openff_evaluator.protocols.analysis), 479 (openff_evaluator.protocols.analysis), 479 (class in attribute), 524 computeResources.CPUTOOLkit (class in copenff_evaluator.backends), 205 attribute), 524 ComputeSymmetryCorrection (class in a openff_evaluator.protocols.paprika.analysis), (openff_evaluator.protocols.yank.BaseYankProtocol 474 (class in attribute), 537 ConcatenateObservables (class in attribute), 537 ConcatenateObservables (class in attribute), 546 openff_evaluator.protocols.paprika.analysis), (openff_evaluator.protocols.yank.LigandReceptorYBinbEatenakeTrajectories (class in attribute), 546 openff_evaluator.protocols.yank.SolvationYankProtocol openff_evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup (class in apenff_evaluator.protocols.yank.SolvationYankProtocol openff_evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup (class in conditions), 560 (openff_evaluator.protocols.groups), 375 method), 570 conditionalGroup.Condition (class in conditions), 570 conditionalGroup.Condition Type (class in conditions), 570 conditionalGroup.Condition (class in conditions), 570 conditionalGroup.Conditions (openff.evaluator.protocols.groups), 375 method), 570 conditionalGroup.Conditions (openff.evaluator.protocols.groups), 570 conditionalGroup.Conditions (openff.evaluator.protocols.grou		-			in
attribute), 364 openff.evaluator.protocols.paprika.analysis), checkpoint_frequency 479 (openff.evaluator.protocols.opennmn.OpenMMSimülutt@Resources (cHoss in attribute), 422 openff.evaluator.backends), 204 checkpoint_frequency ComputeResources.GPITOOLkit (class in (openff.evaluator.protocols.simulation.BaseSimulation openff.evaluator.backends), 205 attribute), 524 ComputeSymmetryCorrection (class in checkpoint_interval openff.evaluator.protocols.paprika.analysis), (openff.evaluator.protocols.yank.BaseYankProtocol 474 attribute), 537 ConcatenateObservables (class in openff.evaluator.protocols.yank.LigandReceptorHiboRzanoateTrajectories (class in checkpoint_interval openff.evaluator.protocols.reweighting), 480 (openff.evaluator.protocols.yank.SlyandReceptorHiboRzanoateTrajectories (class in attribute), 546 openff.evaluator.protocols.reweighting), 485 checkpoint_interval ConditionalGroup (class in (openff.evaluator.protocols.yank.SolvationMarkProtocol openff.evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup.Condition (class in clear() (openff.evaluator.utils.observables.Observables.Observable.openff.evaluator.protocols.groups), 375 method), 570 conditionalGroup.Condition.Type (class in clear_gradientsC) (openff.evaluator.utils.observables.Observable.openff.evaluator.protocols.groups), 375 method), 568 confidence_intervals clear_gradientsC) (openff.evaluator.utils.observables.ObservableAttribute), 304 complex_confiate_path connection_options (class in openff.evaluator.protocols.groups), 375 method), 570 attribute), 304 (openff.evaluator.protocols.paprika.restraints.GenerateAttadthibitartpite), 376 (openff.evaluator.protocols.paprika.restraints.GenerateAttadthibitartpite), 376 (openff.evaluator.protocols.paprika.restraints.GenerateAttadthibitartpite), 376 (openff.evaluator.protocols.paprika.restraints.GenerateAttadthibitartpite), 376 (openff.evaluator.protocols.paprika.restraints.GenerateAttadthibitartpite), 304 (openff.evaluator.protocols.paprika.restraints.GenerateAttadthib					
<pre>checkpoint_frequency 479</pre>				(
(openff.evaluator.protocols.opennm.OpenfMMSimfibinguteResources (lass in attribute), 422 openff.evaluator.backends), 204 checkpoint_frequency ComputeSequetaes.GPUToolkit (class in (openff.evaluator.protocols.simulation.BaseSimulation openff.evaluator.backends), 205 attribute), 524 ComputeSymmetryCorrection (class in checkpoint_interval openff.evaluator.protocols.paprika.analysis), (openff.evaluator.protocols.yank.BaseYankProtocol 474 attribute), 537 ConcatenateObservables (class in openff.evaluator.protocols.yank.LigandReceptorYEOnEPtranet eTrajectories (class in attribute), 536 conditional Group (class in attribute), 546 openff.evaluator.protocols.reweighting), 490 (openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.reweighting), 485 checkpoint_interval ConditionalGroup (class in clear (openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup Condition Type (class in clear (openff.evaluator.protocols.groups), 375 method), 570 conditionalGroup.Condition.Type (class in clear_gradients () (openff.evaluator.utils.observables/Deservables/Deservable@penff.evaluator.protocols.groups), 375 method), 570 conditionalGroup.Condition.Type (class in clear_gradients () (openff.evaluator.utils.observables.ObservableRogenff.evaluator.protocols.groups), 375 method), 570 conditionalGroup.conditionalGroup clear_gradients () (openff.evaluator.utils.observables.ObservableRogenff.evaluator.protocols.analysis.AverageFreeEnergies attribute), 454 connection.ptions (openff.evaluator.client.Request (openff.evaluator.utils.observables.ObservableRogenff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttradtRogenff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttradtRogenff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttradtRogenff.evaluator.client.neonponents.conversion), complex_electrostatic_lambdas 180 (openff.evaluator.protocols.paprik		(openff.evaluator.protoco	ois.paprika.anaiysis),
atribure), 422 checkpoint_frequency computeResources.GPUToolkit (class in (openff.evaluator.protocols.simulation.BaseSimulation openff.evaluator.backends), 204 checkpoint_interval (openff.evaluator.protocols.yank.BaseYankProtocol 474 atribure), 537 ConcatenateObservables (class in checkpoint_interval (openff.evaluator.protocols.yank.BaseYankProtocol 474 atribure), 537 ConcatenateObservables (class in checkpoint_interval (openff.evaluator.protocols.yank.BaseYankProtocol 474 atribure), 546 (openff.evaluator.protocols.yank.LigandReceptorYBobEntonaheTrajectories (class in (openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.reweighting), 490 (openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.reweighting), 495 checkpoint_interval ConditionalGroup (class in (openff.evaluator.utils.observables.ObservableFrame openff.evaluator.protocols.groups), 374 atribute), 554 ConditionalGroup.Condition (class in clear_gradientsQ)(openff.evaluator.utils.observables.ObservableFrame openff.evaluator.protocols.groups), 375 method), 570 ConditionalGroup.Condition (penff.evaluator.utils.observables.ObservableAtribute), 376 method), 567 conditions(openff.evaluator.utils.observables.ObservableAtribute), 376 method), 568 confidence_intervals confidence_interva			+/9	(alaaa	<i></i>
checkpoint_frequency ComputeResources.GPUToolkit (class in (openff.evaluator.protocols.simulation.BaseSimulation openff.evaluator.backends), 205 attribute), 524 ComputeSymmetryCorrection (class in checkpoint_interval openff.evaluator.protocols.paprika.analysis), (openff.evaluator.protocols.yank.BaseYankProtocol 474 attribute), 537 ConcatenateObservables (class in openff.evaluator.protocols.reveighting), 490 (openff.evaluator.protocols.yank.LigandReceptorYEobf2atment eTrajectories (class in attribute), 546 openff.evaluator.protocols.reveighting), 490 (openff.evaluator.protocols.yank.LigandReceptorYEobf2atment eTrajectories (class in attribute), 546 openff.evaluator.protocols.reveighting), 485 (clear) (openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.reveighting), 485 (clear) (openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup (class in clear) (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups), 375 method), 567 ConditionalGroup.Condition.Type (class in clear_gradients() (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups.ConditionalGroup clear_gradients() (openff.evaluator.utils.observables.ObservableAtartipute), 376 method), 568 confidence_intervals clear_gradients() (openff.evaluator.utils.observables.ObservableRopenff.evaluator.protocols.analysis.AverageFreeEnergies method), 570 connection_options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenetaeAttaadHisburgh/82 complex_coordinate_path connection_options (class in openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenetaeAttaadHisburgh/82 complex_coordinate_path so (openff.evaluator.protocols.paprika.cestraints.GenetaeAttaadHisburgh/82 complex_coordinate_path so (openff.evaluator.protocols.paprika.cestraints.GenetaeAttaadHisburgh/82 complex_file_path (openff.evaluator.protocols.paprika.coordinates/PrepreVIL/Continetes attribute),				•	in
(openff.evaluator.protocols.simulation.BaseSimulation openff.evaluator.backends), 205 attribute), 524 ComputeSymmetryCorrection (class in checkpoint_interval openff.evaluator.protocols.yank.BaseYankProtocol 474 attribute), 537 ConcatenateObservables (class in openff.evaluator.protocols.yank.BaseYankProtocol 474 (openff.evaluator.protocols.yank.LigandReceptorYBobExtendateTrajectOries (class in openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.reweighting), 490 (openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup (class in (openff.evaluator.protocols.groups), 375 method), 570 ConditionalGroup.Condition (class in clear_Q(openff.evaluator.utils.observables.ObservableFrame openff.evaluator.protocols.groups), 375 method), 570 ConditionalGroup.Condition.Type (class in clear_gradientsQ) (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups), 375 method), 561 Conditions (openff.evaluator.protocols.groups), 375 method), 560 conditions (openff.evaluator.protocols.groups), 376 method), 560 conditions (openff.evaluator.protocols.analysis.AverageFreeEnergies method), 570 attribute; 304 complex_coordinate_path connection_options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttraidhistant)v82 attribute), 449 Connection_options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttraidhistant)v82 attribute), 455 openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttraidhistant)v82 attribute), 454 coordinates.StoredSimulation. (openff.evaluator.protocols.paprika.cestraints.GenerateMateXtraitesestensityData (class in attribute), 454 coordinates.StoredSimulation.Data attribute), 454 coordinates.StoredSimulation.Data attribute), 454 coordinates.StoredSimulationData attribute), 454 coordinates.StoredSimulationData attribute), 454 coordinates.StoredSimulationD					in
attribute), 524 ComputeSymmetryCorrection (class in openff:evaluator.protocols.yank.BaseYankProtocol 474 attribute), 537 ConcatenateObservables (class in openff:evaluator.protocols.yank.BaseYankProtocol 474 attribute), 537 ConcatenateObservables (class in openff:evaluator.protocols.yank.LigandReceptorYtobeEtrement eTrajectories (class in openff:evaluator.protocols.reweighting), 490 (openff:evaluator.protocols.yank.LigandReceptorYtobeEtrement eTrajectories (class in openff:evaluator.protocols.reweighting), 485 ccheckpoint_interval ConditionalGroup (class in openff:evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup.Condition (class in openff:evaluator.protocols.groups), 375 method), 570 conditionalGroup.Condition.Type (class in openff:evaluator.protocols.groups), 375 method), 560 conditions (openff:evaluator.protocols.groups), 375 (openff:evaluator.utils.observables.ObservableAttribute), 376 complex_coordinate_path connection_options (openff:evaluator.protocols.paprika.restraints.GenerateAttadtHebtraint/2 attribute), 449 Connection_options (class in openff:evaluator.protocols.paprika.restraints.GenerateAttadtHebtraint/2 attribute), 440 Connection_options (class in openff:evaluator.client.Request (openff:evaluator.protocols.paprika.restraints.Genera					in
checkpoint_interval openff.evaluator.protocols.paprika.analysis), (openff.evaluator.protocols.yank.BaseYankProtocol 474 Atribute), 537 ConcatenateObservables (class in openff.evaluator.protocols.yank.ligandReceptorYEobEautenateTrajectories (class in openff.evaluator.protocols.yank.ligandReceptorYEobEautenateTrajectories (class in (openff.evaluator.protocols.yank.ligandReceptorYEobEautenateTrajectories) (class in (openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.groups), 374 (trainel, 554 ConditionalGroup Condition (class in (clear Q (openff.evaluator.utils.observables.ObservableFrame openff.evaluator.protocols.groups), 375 method), 570 ConditionalGroup. Condition (rype (class in clear_gradientsC) (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups), 375 method), 567 conditions (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups), 375 method), 568 confidence_intervals clear_gradientsC) (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups), 375 method), 570 conditions (openff.evaluator.protocols.groups). 375 method), 576 conditions (openff.evaluator.protocols.groups), 375 method), 570 conditions (openff.evaluator.protocols.groups). 600 clear_gradientsC) (openff.evaluator.utils.observables.ObservableRometievals clear_gradientsC) (openff.evaluator.utils.observables.ObservableRometievals clear_gradientsC) (openff.evaluator.utils.observables.ObservableRometievals clear_gradientsC) (openff.evaluator.protocols.paprika.restraints.GenerateAttadthbikmthyt82 attribute), 449 Connection_options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttadthbikmthyt82 attribute), 445 openff.evaluator.protocols.paprika.coordinates/Preptifevaluator.datasets.curation.components.conversion), complex_file_path (openff.evaluator.protocols.paprika.coordinates/Preptifevaluator.datasets.curation.components.conversion), complex_file_path (openff.evaluator.protoco					in
(openff.evaluator.protocols.yank.BaseYankProtocol 474 attribute), 537 ConcatenateObservables (class in openff.evaluator.protocols.yank.LigandReceptorYEiokPatenakeTrajectories (class in attribute), 546 openff.evaluator.protocols.reveighting), 485 checkpoint_interval ConditionalGroup (class in (openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup (class in clear_G(openff.evaluator.utils.observables.ObservableFrame openff.evaluator.protocols.groups), 375 method), 570 conditionalGroup.Condition.Type (class in clear_gradients O (openff.evaluator.utils.observables.ObservableAurribure), 376 conditions (openff.evaluator.protocols.groups), 375 complex_coordinate_path connection_options (openff.evaluator.client.Request (openff.evaluator.utils.observables.ObservableAurribure), 304 constitions (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttaadtNistardyn%2 attribute), 449 connection_options (class in copenff.evaluator.protocols.paprika.restraints.GenerateAttaadtNistardyn%2 attribute, 455 openff.evaluator.client.Reques		-		· · · · · · · · · · · · · · · · · · ·	
airibue), 537 ConcatenateObservables (class in checkpoint_interval openff:evaluator.protocols.reweighting), 490 (openff:evaluator.protocols.yank.LigandReceptorYEnbRatmenkeTrajectories (class in attribute), 546 conditionalGroup (class in (openff:evaluator.protocols.yank.SolvationYankProtocol openff:evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup (class in Clear Q (openff:evaluator.protocols.yank.SolvationYankProtocol openff:evaluator.protocols.groups), 375 method), 550 ConditionalGroup. Condition (class in Clear_gradients () (openff:evaluator.utils.observables.Observable openff:evaluator.protocols.groups), 375 method), 567 ConditionalGroup.Condition_Type (class in Clear_gradients () (openff:evaluator.utils.observables.Observable openff:evaluator.protocols.groups), 375 method), 568 confidence_intervals ConditionalGroup.Condition.Type (class in Clear_gradients () (openff:evaluator.utils.observables.ObservableAutripute), 376 method), 560 confidence_intervals Clear_gradients () (openff:evaluator.utils.observables.ObservableAutripute), 376 method), 560 confidence_intervals Connection_options (openff:evaluator.client.Request (openff:evaluator.protocols.paprika.restraints.GenerateAttaddMburthitk2 attribute), 449 ConnectionOptions (class in openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttaddMburthitk2 attribute), 455 openfika.restraints.GenerateAttaddMburthitk2 (openff.evaluator.protocols.paprika.restraints.GenerateAttaddMburthitk2 (openff.evaluator.protocols.paprika.restraints.GenerateAttaddMburthitk2 (openff.evaluator.protocols.paprika.restraints.GenerateAttaddMburthitk2 (openff.evaluator.protocols.paprika.restraints.GenerateAttaddMburthitk2 (openff.evaluator.protocols.paprika.coordinates/MemperPullCoordinates in attribute), 455 conservice of StabilityData (class in attribute), 454 coordinates 180 (openff.evaluator.protocols.paprika.coordinates/MemperPullCoordinates attribute), 454 coordinates/MemperPullCoordinates attribute), 4				ns.paprika.anaiysis),
checkpoint_interval openff.evaluator.protocols.reweighting), 490 (openff.evaluator.protocols.yank.LigandReceptorYibikPattenakeTrajectories (class in attribute), 546 openff.evaluator.protocols.groups), 374 (openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup (class in (openff.evaluator.protocols.groups), 375 method), 570 ConditionalGroup.Condition.Type (class in clear_gradientsQ) (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups), 375 method), 567 conditions (openff.evaluator.protocols.groups), 375 method), 567 conditions (openff.evaluator.protocols.groups), 376 method), 568 confidence_intervals Clear_gradientsQ) (openff.evaluator.utils.observables.ObservableAttribute), 376 method), 568 confidence_intervals Clear_gradientsQ) (openff.evaluator.utils.observables.ObservableAttribute), 376 method), 570 conditions (openff.evaluator.protocols.groups.ConditionalGroup Clear_gradientsQ) (openff.evaluator.utils.observables.ObservableAttribute), 376 method), 570 conditions (openff.evaluator.client.Request (openff.evaluator.utils.observables.ObservableAttribute), 304 complex_coordinate_path connection_options (logenff.evaluator.client.Request (openff.evaluator.client.Request (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttadtMaturbity%2 attribute), 449 ConnectionOptions (class in openff.evaluator.client), complex_coordinate_path 80 (openff.evaluator.protocols.paprika.restraints.GenerateAttadtMaturbity%2 (openff.evaluator.protocols.paprika.restraints.GenerateAttadtMaturbity%2 (openff.evaluator.protocols.paprika.coordinatesAPPeparePullCoordinates attribute), 434 openfi.evaluator.datasets.curation.components.conversion), complex_file_path (openff.evaluator.protocols.paprika.coordinatesAPPeparePullCoordinates attribute), 440 attribute), 229 complex_steric_lambdas coordinatesAPPeparePullCoordinates.BuildCoordinatesPackmo attribute), 544 attribute, 331 Component (class in openff.eval				(class	in
(openff.evaluator.protocols.yank.LigandReceptorYiob2RumonteTrajectories (class in attribute), 546 openff.evaluator.protocols.reweighting), 485 checkpoint_interval ConditionalGroup (class in (openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup.Condition (class in clear() (openff.evaluator.utils.observables.ObservableFrame openff.evaluator.protocols.groups), 375 method), 570 ConditionalGroup.Condition.Type (class in clear_gradients() (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups), 375 method), 567 conditions(openff.evaluator.protocols.groups), 375 method), 568 confidence_intervals clear_gradients() (openff.evaluator.utils.observables.ObservableAttribute), 376 method), 570 conditionalGroup clear_gradients() (openff.evaluator.utils.observables.ObservableAttribute), 376 method), 570 confidence_intervals clear_gradients() (openff.evaluator.utils.observables.ObservableKopmeff.evaluator.protocols.analysis.AverageFreeEnergies method), 570 attribute), 304 complex_coordinate_path connection_options (class in openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttaadtbistrainuts2 attribute), 449 Connection_options (class in openff.evaluator.client), complex_coordinate_path 80 (openff.evaluator.protocols.paprika.restraints.GeneCartPutilResembBensityData (class in attribute), 455 openfi.evaluator.datasets.curation.components.conversion), complex_electrostatic_lambdas 180 (openff.evaluator.protocols.paprika.coordinates/PreparePullCoordinates attribute), 434 coordinates/PreparePullCoordinates attribute), 434 coordinates/PreparePullCoordinates attribute), 434 coordinates/PreparePullCoordinates attribute), 434 coordinates/PreparePullCoordinates/PreparePullCoordinates/Packmod attribute), 531 Component (class in openff.evaluator.protocols.paprika.coordinates/PreparePullCoordinates.BuildCoordinatesPackmo attribute), 54 attribute, 331 Component (class in openff.evaluator.protocols.yank.Liga					
attribute), 546 openff:evaluator.protocols.reweighting), 485 checkopint_interval ConditionalGroup (class in (openff:evaluator.protocols.yank.SolvationYankProtocol openff:evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup.Condition (class in Clear() (openff:evaluator.utils.observables.ObservableFrame openff:evaluator.protocols.groups), 375 method), 570 ConditionalGroup.Condition.Type (class in Clear_gradients() (openff:evaluator.utils.observables.Observable openff:evaluator.protocols.groups.ConditionalGroup class in clear_gradients() (openff:evaluator.utils.observables.ObservableAurribute), 376 conditions (openff:evaluator.protocols.groups.ConditionalGroup clear_gradients() (openff:evaluator.utils.observables.ObservableKoppenff:evaluator.protocols.analysis.AverageFreeEnergies method), 568 confidence_intervals clear_gradients() (openff:evaluator.utils.observables.ObservableKoppenff:evaluator.client.Request (openff:evaluator.protocols.paprika.restraints.GenerateAttadtMbistatift%22 attribute), 304 complex_coordinate_path connectionOptions (class in openff:evaluator.client.Request (openff:evaluator.protocols.paprika.restraints.GenerateAttadtMbistatift%2 in complex_coordinate_path 80 openeff:evaluator.datasets.curation.components.conversion),					
checkpoint_interval ConditionalGroup (class in (openff_evaluator.protocols.yank.SolvationYankProtocol openff_evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup.Condition (class in clear() (openff_evaluator.utils.observables.ObservableFrame openff_evaluator.protocols.groups), 375 in method), 570 ConditionalGroup.Condition.Type (class in clear_gradients() (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups), 375 in in clear_gradients() (openff.evaluator.utils.observables.ObservableAurripute), 376 confitions (openff.evaluator.protocols.analysis.AverageFreeEnergies method), 570 confitence_intervals confitence_intervals clear_gradients() (openff.evaluator.utils.observables.ObservableAurripute), 304 connection_options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttadhMetaripM2 attribute), 449 connection_options (class in openff.evaluator.client), 800 (openff.evaluator.protocols.paprika.restraints.GenerateAttadhMetaripM2 attribute, 449 connection_options (class in openff.evaluator.client), 800 (openff.evaluator.protocols.paprika.cestraints.GenerateAttadhMetaripM2 constitutor.clients in (openff.evaluator.				•	
<pre>(openff.evaluator.protocols.yank.SolvationYankProtocol openff.evaluator.protocols.groups), 374 attribute), 554 ConditionalGroup.Condition (class in clear() (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups), 375 method), 570 ConditionalGroup.Condition.Type (class in clear_gradients() (openff.evaluator.utils.observables.Observables.Observable openff.evaluator.protocols.groups), 375 method), 567 clear_gradients() (openff.evaluator.utils.observables.Observables.Observables.openff.evaluator.protocols.groups.ConditionalGroup clear_gradients() (openff.evaluator.utils.observables.Observables.Observables.openff.evaluator.protocols.groups.ConditionalGroup clear_gradients() (openff.evaluator.utils.observables.Observables.Observables.openff.evaluator.protocols.analysis.AverageFreeEnergies method), 568 confidence_intervals clear_gradients() (openff.evaluator.utils.observables.Observables.Openff.evaluator.protocols.analysis.AverageFreeEnergies method), 570 condition_options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttaddWatstation%2 attribute), 449 connection_options (olass in openff.evaluator.client), complex_coordinate_path</pre>					
attribute), 554 ConditionalGroup.Condition (class in openff.evaluator.protocols.groups), 375 clear() (openff.evaluator.utils.observables.Observables Prame openff.evaluator.protocols.groups), 375 in clear_gradients() (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups), 375 in method), 567 conditions (openff.evaluator.protocols.groups), 375 method), 568 conditions (openff.evaluator.protocols.groups). 376 method), 568 confidence_intervals clear_gradients() (openff.evaluator.utils.observables.Observable&Attribute), 304 connection_options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttadMwishtribute), 304 complex_coordinate_path connectionOptions (class in openff.evaluator.client), 80 complex_coordinate_path 80 (openff.evaluator.protocols.paprika.restraints.Gen@attribute), 449 connectionOptions (class in openff.evaluator.client), 80 complex_electrostatic_lambdas 180 (openff.evaluator.protocols.paprika.restraints.Gen@attributeBatessbensityData (class in openff.evaluator.datasets.curation.components.conversion), complex_electrostatic_lambdas complex_file_path (openff.evaluator.protocols.paprika.coordinates.BP@parePullCoordinates attribute), 544 coordinate_file_name coordinate_file_path (openff.evaluator.protocols.paprika.coordinates.APreptirekulator.datasets.curation.componen	-			•	in
<pre>clear() (openff.evaluator.utils.observables.ObservableFrame openff.evaluator.protocols.groups), 375 method), 570</pre>					in
method), 570 ConditionalGroup.Condition.Type (class in Clear_gradients() (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups), 375 method), 567 Clear_gradients() (openff.evaluator.utils.observables.Observable/Autrigiute), 376 method), 568 confidence_intervals Clear_gradients() (openff.evaluator.utils.observables.Observable/Autrigiute), 376 method), 568 confidence_intervals Clear_gradients() (openff.evaluator.utils.observables.Observable/Repression.got(), 304 consection.options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttadHwisutatit%2 attribute), 449 ConnectionOptions (class in openff.evaluator.client), complex_coordinate_path 80 openff.evaluator.client), consection.components.conversion), complex_electrostatic_lambdas 180 openff.evaluator.datasets.curation.components.conversion), complex_file_path openff.evaluator.protocols.yank.LigandReceptorYioHintExcessDensityDataSchema (class in openff.evaluator.protocols.paprika.coordinates/PreparePullCoordinates complex_file_path openff.evaluator.datasets.curation.components.conversion), complex_steric_lambdas 180 (openff.evaluator.protocols.yank.LigandReceptorYioHintExcessDensityDataSchema (class in openff.evaluator.protocols.paprika.			_		in
clear_gradients() (openff.evaluator.utils.observables.Observable openff.evaluator.protocols.groups), 375 method), 567 conditions (openff.evaluator.protocols.groups.ConditionalGroup clear_gradients() (openff.evaluator.utils.observables.ObservableAutripute), 376 method), 568 confidence_intervals clear_gradients() (openff.evaluator.utils.observables.ObservableAutripute), 304 complex_coordinate_path connection_options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GeneratAttadHibiarufs2) attribute), 449 connection_options (class in openff.evaluator.client), complex_coordinate_path 80 (openff.evaluator.protocols.paprika.restraints.GeneratAttadHibiarufs2) attribute), 455 openff.evaluator.client.components.conversion), complex_ecoordinate_path 80 (openff.evaluator.protocols.paprika.restraints.GeneratAttadHibiarufs2) attribute), 455 openff.evaluator.datasets.curation.components.conversion), complex_electrostatic_lambdas 180 (openff.evaluator.protocols.yank.LigandReceptorYiohRetrEacessDensityDataSchema (class in attribute), 434 coordinates attribute), 434 coordinates/PreparePullCoordinates attribute), 434 coordinates/PreparePullCoordinates attribute), 434 coordinates/PreparePullCoordinates attribute), 440 attribute), 229 complex_steric_lambdas coordinates/file_path (openff.evaluator.protocols.yank.LigandReceptorYiohRetrEacessDensityDataSchema (class in attribute), 440 attribute), 229 complex_steric_lambdas coordinates/file_path (openff.evaluator.protocols.yank.LigandReceptorYinkProtocdupenff.evaluator.protocols.coordinatesBuildCoord					in
method), 567 conditions (openff. evaluator.protocols.groups. ConditionalGroup Clear_gradients() (openff.evaluator.utils.observables.ObservablesArtripute), 376 method), 568 clear_gradients() (openff.evaluator.utils.observables.ObservablesAntripute), 376 confidence_intervals clear_gradients() (openff.evaluator.utils.observables.ObservablesAntripute), 304 consection_options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttadtRikkminit%2 complex_coordinate_path connectionOptions (class in openff.evaluator.client), 80 complex_coordinate_path 80 (openff.evaluator.protocols.paprika.restraints.GenerateAttadtRikkminit%2 attribute), 455 consectionOptions (class in openff.evaluator.client), 80 so (openff.evaluator.protocols.paprika.restraints.Gen@envPnffExsesbDensityData (class in attribute), 455 openff.evaluator.dtasets.curation.components.conversion), complex_electrostatic_lambdas complex_electrostatic_lambdas 180 (openff.evaluator.protocols.paprika.coordinates/PreparePullCoordinates complex_file_path (openff.evaluator.protocols.paprika.coordinates/PreparePullCoordinates complex_file_path (openff.evaluator.protocols.paprika.coordinates/PreparePullCoordinates complex_steric_lambdas coordinate_file_name complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols			_		
clear_gradients() (openff.evaluator.utils.observables.ObservableAutribute), 376 method), 568 confidence_intervals clear_gradients() (openff.evaluator.utils.observables.ObservableKippereff.evaluator.protocols.analysis.AverageFreeEnergies method), 570 attribute), 304 complex_coordinate_path connection_options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttadtReistuta)(%2 attribute), 449 connectionOptions (class in openff.evaluator.client), complex_coordinate_path %0 (openff.evaluator.protocols.paprika.restraints.GenerateAttadtReistuta)(%2 attribute), 455 openff.evaluator.protocols.paprika.restraints.GenerateAttadtReistuta)(%2 attribute), 455 openff.evaluator.protocols.paprika.restraints.GenerateAttadtReistuta)(%2 attribute), 455 openff.evaluator.protocols.paprika.restraints.GenerateAttadtReistuta)(%2 complex_coordinate_path %0 (openff.evaluator.protocols.paprika.restraints.GenerateAttadtReistuta)(%2 attribute), 455 openff.evaluator.protocols.paprika.restraints.GenerateAttadtReistestesteners.curation.components.conversion), complex_electrostatic_lambdas 180 (openff.evaluator.protocols.yank.LigandReceptorYiohRentbEacessDensityDataSchema (class in attribute), 544 openff.evaluator.protocols.paprika.coordinates/PreparePullCoordinates attribute), 440 coordinates/PreparePullCoordinates attribute), 440 coordinates/ApreptiveRulataveGimediatexe.StoredSimulationData attribute), 440 attribute), 229 complex_steric_lambdas coordinates/ApreptiveRulataveGimediatexe.StoredSimulationData attribute), 544 attribute), 327					litionalGroup
method), 568 confidence_intervals clear_gradients() (openff.evaluator.utils.observables.Observables.Components.Conversion), complex_electrostatic_lambdas complex_clectrostatic_lambdas Condinate_file_name <td< td=""><td></td><td></td><td></td><td></td><td>onur or onp</td></td<>					onur or onp
clear_gradients() (openff.evaluator.utils.observables.Observables.Observables.Complex_coordinate_path connection_options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttadtkisturatives.2 attribute), 449 ConnectionOptions (class in openff.evaluator.client), complex_coordinate_path 80 (openff.evaluator.protocols.paprika.restraints.GenerateAttadtkisturatives.Class in openff.evaluator.client), complex_electrostatic_lambdas 180 (openff.evaluator.protocols.yank.LigandReceptor YiothPentDeakessDensityData (class in attribute), 454 openff.evaluator.datasets.curation.components.conversion), complex_file_path (openff.evaluator.protocols.paprika.coordinates.Protocols.paprika.coordinatesProtocols.paprika.coordinatesProtocols.paprika.coordinatesProtocols.paprika.coordinatesProtocols.paprika.coordinatesProtocols.paprika.coordinatesProtocols.paprika.coordinatesProtocols.coordinates.BuildCoordinatesPackmo attribute), 434 coordinate_file_name complex_file_path (openff.evaluator.protocols.paprika.coordinatesApreffireRdilauseGtoradje.adata.StoredSimulationData attribute), 440 attribute), 229 complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtocalpenff.evaluator.protocols.coordinatesPackmo attribute), 544 coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtocalpenff.evaluator.protocols.coordinatesPackmo attribute), 544 coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtocalpenff.evaluator.protocols.coordinatesPackmo attribute), 544 coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtocalpenff.evaluator.protocols.coordinatesBuildCoordinatesPackmo attribute), 544 coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtocalpenff.evaluator.protocols.coordinates.BuildCoordinatesPackmo attribute), 544 coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtocalpenff.evaluator.protocols.coordinates.BuildCoordinatesPackmo attribute), 544 c					
method), 570 attribute), 304 complex_coordinate_path connection_options (openfi.evaluator.client.Request (openfi.evaluator.protocols.paprika.restraints.GenerateAttadtRibibute), tf82 attribute), 449 ConnectionOptions (class in openfi.evaluator.client), complex_coordinate_path 80 (openfi.evaluator.protocols.paprika.restraints.Gen@conv@full&sestebDensityData (class in openfi.evaluator.client), complex_coordinate_path 80 (class in openfi.evaluator.components.conversion), complex_electrostatic_lambdas 180 (openfi.evaluator.protocols.yank.LigandReceptorYEin&PentEakcessDensityDataSchema (class in openfi.evaluator.protocols.paprika.coordinates/PreparePullCoordinates complex_file_path (openfi.evaluator.protocols.paprika.coordinates/PreparePullCoordinates StoredSimulationData attribute), 229 complex_steric_lambdas coordinate_file_path (openfi.evaluator.protocols.yank.LigandReceptorYankProtocdpenfi.evaluator.protocols.coordinates.BuildCoordinatesPackmo attribute), 544 attribute), 331				ols.analysis.Average	eFreeEnergies
complex_coordinate_path connection_options (openff.evaluator.client.Request (openff.evaluator.protocols.paprika.restraints.GenerateAttaadtNiskntai)t%2 attribute), 449 ConnectionOptions (class in openff.evaluator.client), complex_coordinate_path 80 (openff.evaluator.protocols.paprika.restraints.GeneConvPrtifKestresbDensityData (class in attribute), 455 openff.evaluator.datasets.curation.components.conversion), complex_electrostatic_lambdas 180 (openff.evaluator.protocols.yank.LigandReceptorYGohPentrEakcessDensityDataSchema (class in attribute), 544 openff.evaluator.datasets.curation.components.conversion), complex_file_path (openff.evaluator.protocols.paprika.coordinateslPPeparePullCoordinates attribute), 434 coordinate_file_name complex_file_path (openff.evaluator.protocols.paprika.coordinateslPPeparePullCoordinates StoredSimulationData attribute), 434 coordinate_file_name coordinate_file_path complex_steric_lambdas coordinate_file_path Coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtocols.coordinates.StoredSimulationData attribute), 544 complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtocols.coordinates.BuildCoordinatesPackmo attribute), 544 attribute				2 0	0
(openff.evaluator.protocols.paprika.restraints.GenerateAttaadtRiikutai)ut%2 attribute), 449 ConnectionOptions (class in openff.evaluator.client), complex_coordinate_path 80 (openff.evaluator.protocols.paprika.restraints.Gen@conte@thtResers#DensityData (class in attribute), 455 complex_electrostatic_lambdas 180 (openff.evaluator.protocols.yank.LigandReceptorYiohRestDensityDataSchema (class in attribute), 544 openff.evaluator.datasets.curation.components.conversion), coordinates complex_file_path (openff.evaluator.protocols.paprika.coordinates!P%pearePullCoordinates attribute), 434 coordinate_file_name coordinates@preffreeRuleascoStoradja.atexta.StoredSimulationData attribute), 440 attribute), 229 complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtocdupenff.evaluator.protocols.coordinatesPackmo attribute), 331 Component (class in openff.evaluator.substances), 129 coordinate_file_path Component (class in openff.evaluator.substances), 129 coordinate_file_path Component (openff.evaluator.protocols.miscellaneous.WeightByMolef(apatifievaluator.protocols.coordinates.SolvateExistingStructure attribute), 400 attribute), 400 attribute), 337	complex_coordinate_path			evaluator.client.Req	uest
attribute), 449 ConnectionOptions (class in openff.evaluator.client), complex_coordinate_path 80 (openff.evaluator.protocols.paprika.restraints.Gen@onvPullRescessiblensityData (class in attribute), 455 complex_electrostatic_lambdas 180 (openff.evaluator.protocols.yank.LigandReceptorYiohRentrEakcessDensityDataSchema (class in attribute), 544 complex_file_path openff.evaluator.datasets.curation.components.conversion), ccomplex_file_path (openff.evaluator.protocols.yank.LigandReceptorYiohRentrEakcessDensityDataSchema ccomplex_file_path (openff.evaluator.protocols.paprika.coordinatesIP%penrePullCoordinates attribute), 434 coordinatesIP%penrePullCoordinates complex_file_path (openff.evaluator.protocols.paprika.coordinatesIP%penrePullCoordinates attribute), 434 coordinate_file_name complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols.paprika.coordinates/Openff.evaluator.protocols.coordinatesPackmon attribute), 229 complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtoc/depenff.evaluator.protocols.coordinatesPackmon attribute), 331 Component (class in openff.evaluator.substances), 129 coordinate_file_path component (openff.evaluator.protocols.miscellaneou					
complex_coordinate_path 80 (openff.evaluator.protocols.paprika.restraints.Gen@onvPntlKesessiblensityData (class in attribute), 455 openff.evaluator.datasets.curation.components.conversion), 0 complex_electrostatic_lambdas 180 (openff.evaluator.protocols.yank.LigandReceptorYionkPentbEak essDensityDataSchema (class in attribute), 544 openff.evaluator.datasets.curation.components.conversion), 0 complex_file_path (openff.evaluator.protocols.paprika.coordinateslP%eparePullCoordinates 0 complex_steric_lambdas coordinate_file_name complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtoc@penff.evaluator.protocols.coordinatesPackmo attribute), 544 attribute), 331 Component (class in openff.evaluator.substances), 129 coordinate_file_path component (openff.evaluator.protocols.miscellaneous.WeightByMolef(rpatimevaluator.protocols.coordin				openff.evaluator.clie	ent),
attribute), 455 openff.evaluator.datasets.curation.components.conversion), complex_electrostatic_lambdas 180 (openff.evaluator.protocols.yank.LigandReceptorYionRentbEacessDensityDataSchema (class in attribute), 544 openff.evaluator.datasets.curation.components.conversion), complex_file_path (openff.evaluator.protocols.paprika.coordinatesIPveparePullCoordinates attribute), 434 coordinate_file_name coordinates(d)prenff.evaluator.StoredSimulationData attribute), 440 attribute), 229 complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtoc@penff.evaluator.protocols.coordinatesPackmo attribute), 544 attribute), 331 Component (class in openff.evaluator.substances), 129 coordinate_file_path component (openff.evaluator.protocols.miscellaneous.WeightByMolef(mpathfievaluator.protocols.coordinates.SolvateExistingStructure attribute), 400 attribute), 337	complex_coordinate_path	1	80		
complex_electrostatic_lambdas 180 (openff.evaluator.protocols.yank.LigandReceptorYGobRentbEakcessDensityDataSchema (class in attribute), 544 openff.evaluator.datasets.curation.components.conversion), complex_file_path (openff.evaluator.protocols.paprika.coordinateslPveparePullCoordinates attribute), 434 complex_file_path (openff.evaluator.protocols.paprika.coordinates(OpenffreeRdlaatseGtandleeAtase.StoredSimulationData attribute), 440 attribute), 229 complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtoc(alpenff.evaluator.protocols.coordinates.BuildCoordinatesPackmo attribute), 544 attribute), 331 Component (class in openff.evaluator.substances), 129 coordinate_file_path component (openff.evaluator.protocols.miscellaneous.WeightByMolef(apatifievaluator.protocols.coordinates.SolvateExistingStructure attribute), 400 attribute), 337	(openff.evaluator.protocols.paprika.restraints.Ge	en &onv&nt/E	& cessiDe nsityData	(class	in
(openff.evaluator.protocols.yank.LigandReceptorYiohRentbExtcessDensityDataSchema (class in attribute), 544 openff.evaluator.datasets.curation.components.conversion), complex_file_path (openff.evaluator.protocols.paprika.coordinateslP%peparePullCoordinates attribute), 434 coordinate_file_name complex_file_path (openff.evaluator.protocols.paprika.coordinates(Dpepff)reRtleatseGtonationAttribute), 434 coordinate_file_name complex_file_path (openff.evaluator.protocols.paprika.coordinates(Dpepff)reRtleatseGtonationAttribute), 440 attribute), 229 complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtoc(atpenff.evaluator.protocols.coordinates.BuildCoordinatesPackmonattribute), 544 attribute), 331 Component (class in openff.evaluator.substances), 129 coordinate_file_path component (openff.evaluator.protocols.miscellaneous.WeightByMolef(rpath@fivevaluator.protocols.coordinates.SolvateExistingStructure attribute), 400 attribute), 337	attribute), 455	(openff.evaluator.dataset	ts.curation.compone	nts.conversion),
attribute), 544 openff.evaluator.datasets.curation.components.conversion), complex_file_path (openff.evaluator.protocols.paprika.coordinateslP%eparePullCoordinates attribute), 434 coordinate_file_name complex_file_path (openff.evaluator.protocols.paprika.coordinates(@pepffrewRdleatseGStonalinates.StoredSimulationData attribute), 440 attribute), 229 complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtoc@penff.evaluator.protocols.coordinates.BuildCoordinatesPackmon attribute), 544 attribute), 331 Component (class in openff.evaluator.substances), 129 coordinate_file_path component (openff.evaluator.protocols.miscellaneous.WeightByMolefCrapetifievaluator.protocols.coordinates.SolvateExistingStructure attribute), 400 attribute), 337	<pre>complex_electrostatic_lambdas</pre>		180		
<pre>complex_file_path (openff.evaluator.protocols.paprika.coordinateslP%eparePullCoordinates</pre>	(openff.evaluator.protocols.yank.LigandReceptor)	·YaokRente	<i>al</i> cessDensityDataSc	chema (class	in
attribute), 434 coordinate_file_name complex_file_path (openff.evaluator.protocols.paprika.coordinates@preffireRdlaatsaGstonalignatumes.StoredSimulationData attribute), 440 attribute), 229 complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtoc@penff.evaluator.protocols.coordinates.BuildCoordinatesPackmon attribute), 544 attribute), 331 Component (class in openff.evaluator.substances), 129 coordinate_file_path component (openff.evaluator.protocols.miscellaneous.WeightByMolef[rapainff.evaluator.protocols.coordinates.SolvateExistingStructure attribute), 400 attribute), 337	attribute), 544	(openff.evaluator.dataset	s.curation.compone	nts.conversion),
<pre>complex_file_path (openff.evaluator.protocols.paprika.coordinates(DpenffmeRdluateseCstanakje.abatea.StoredSimulationData</pre>	$\verb complex_file_path (openff.evaluator.protocols.paprika.evaluator.protocols.papri$	coordinates	PeparePullCoordinate	es	
attribute), 440 attribute), 229 complex_steric_lambdas coordinate_file_path (openff.evaluator.protocols.yank.LigandReceptorYankProtoc@penff.evaluator.protocols.coordinates.BuildCoordinatesPackmon attribute), 544 attribute), 331 Component (class in openff.evaluator.substances), 129 coordinate_file_path component (openff.evaluator.protocols.miscellaneous.WeightByMolefinpervaluator.protocols.coordinates.SolvateExistingStructure attribute), 337					
<pre>complex_steric_lambdas coordinate_file_path</pre>	$\verb complex_file_path (openff.evaluator.protocols.paprika.evaluator.protocols.papri$	coordinates	. ØprepfineRdleatse Gtonalj	e.alat a.StoredSimula	tionData
(openff.evaluator.protocols.yank.LigandReceptorYankProtoc@penff.evaluator.protocols.coordinates.BuildCoordinatesPackmo attribute), 544 Component (class in openff.evaluator.substances), 129 coordinate_file_path component (openff.evaluator.protocols.miscellaneous.WeightByMolef(rpanifievaluator.protocols.coordinates.SolvateExistingStructure attribute), 337			· · · · · · · · · · · · · · · · · · ·		
attribute), 544 attribute), 331 Component (class in openff.evaluator.substances), 129 coordinate_file_path component (openff.evaluator.protocols.miscellaneous.WeightByMole [repainffievaluator.protocols.coordinates.SolvateExistingStructure attribute), 400 attribute), 337	complex_steric_lambdas				
<pre>component (openff.evaluator.protocols.miscellaneous.WeightByMoleKapetiofhevaluator.protocols.coordinates.SolvateExistingStructure</pre>				cols.coordinates.Bui	ldCoordinatesPackmo
component (openff.evaluator.protocols.miscellaneous.WeightByMoleKrapetiofhevaluator.protocols.coordinates.SolvateExistingStructure attribute), 400 attribute), 337	Component (class in openff.evaluator.substances), 129				
	$\verb component (open ff. evaluator. protocols. miscellaneous. Weight the second $	ghtByMole H	apetiofnevaluator.protoc	cols.coordinates.Solv	vateExistingStructure
component.Role (<i>class in open)</i> , evaluator.substances), coordinate_fife_path	Component.Role (class in openff.evaluator.substances),		te_file_path		

		vtocols.forcefield.Bas	eBuildSy	ydaata_cl	ass() (openff.evaluator.s	storage.query.H	FreeEnergyDataQuery
	attribute), 348				class method), 245		
coordin	ate_file_path				ass() (openff.evaluator.s	storage.query.S	imulationDataQuery
	(openff.evaluator.pro	vtocols.forcefield.Bui	ldLigPar				
	attribute), 359			data_to	_store (<i>openff.evaluator</i>	layers.Calculo:	ationLayerResult
coordin	ate_file_path				attribute), 190		
	(openff.evaluator.pro	vtocols.forcefield.Bui	ldSmirne	ATALENO.	_store (openff.evaluator	workflow.Work	kflowResult
	attribute), 354				attribute), 260		
coordin	ate_file_path			Decorre	lateObservables	(class	in
	(openff.evaluator.pro	vtocols.forcefield.Bui	ldTLeap	System	openff.evaluator.protoco	ls.analysis), 32	24
	attribute), 365		-	Decorre	lateTrajectory	(class	in
coordin	ate_file_path				openff.evaluator.protoco	ls.analysis), 31	8
	(openff.evaluator.pro	otocols.storage.Unpa	ckStored	Sienfaulit	pamparika_schema()	•	
	attribute), 531	Ŭ I			(openff.evaluator.proper	ties.HostGuest	BindingAffinity
copy()	, .	workflow.utils.Protoc	olPath		class method), 122		0 30 2
	method), 285	0		default.	_request_options()		
count()	, .	storage.attributes.Fi	lePath		(openff.evaluator.client.l	EvaluatorClien	t static
	method), 249				method), 78		
count e		f.evaluator.protocols	.coordin	ale£Bu]ltl	CrewdivghasiRugkozhema	0	
	attribute), 330				(openff.evaluator.proper		static
count e	<pre>xact_amount(openfl</pre>	f.evaluator.protocols	.coordin	ates.Solva			
	attribute), 337				_reweighting_schema	0	
Curatio	nComponent	(class	in		(openff.evaluator.proper		Constant
curucio	openff.evaluator.data				static method), 109	nes.Dieteente	sonstant
	152	iseis.euraiton.eompo	nems),	default	_reweighting_schema	\cap	
Curatio	nComponentSchema	(class	in	ucruurt.	(openff.evaluator.proper		fMixino
curucio	openff.evaluator.data	•			class method), 112	iies.EninuipyO	Junixing
	153	iseis.euraiton.eompo	nems),	default	_reweighting_schema	\cap	
Curatio	nWorkflow	(class	in	ucruurt.	(openff.evaluator.proper		Waporization
curacio	openff.evaluator.data	·			class method), 116	ites.EninuipyO	jvaponzanon
	153	iseis.curation.workja	<i>JW</i>),	dofaul+	_reweighting_schema	0	
Curatio	nWorkflowSchema	(class	in	uerauri.	(openff.evaluator.proper		larVolume
Curatio	openff.evaluator.data				class method), 105	nes.Excession	urvolume
	154	iseis.curation.workja		dofaul+	_simulation_schema()	`	
current		avaluator protocols (Gapenoff.evaluator.proper		static
current	attribute), 376	evaluator.protocois.g	groups.C	οπαιτισπαι	method), 101	iles.Densily	siunc
cutoff(, .	fold Lie DayCon For	- Fields	പക്രവ]+	_simulation_schema()	`	
cutorr (Jiela.LigrarGenford	rerieius				Constant
	property), 186	fold TL om Four oFic	ldCourse		(openff.evaluator.proper	lies.Dielectrico	Jonsiani
cutorr (openff.evaluator.force property), 184	јена.Теаргонсегне	usource		static method), 108	`	
	property), 164			uerauri.	_simulation_schema() (openff.evaluator.proper		Mining
D						iles.EninaipyO	JMIXINg
				dafaul+	class method), 112	`	
DaskLoc	alCluster	(class	in	derault.	_simulation_schema()		Alan anis ati an
	openff.evaluator.back				(openff.evaluator.proper	ties.EnthalpyO	Jvaporization
DaskLSF	Backend	(class	in	1. (static method), 115	、 、	
	openff.evaluator.back			default.	_simulation_schema()		1 37 1
DaskPBS	Backend	(class	in		(openff.evaluator.proper	ties.ExcessMol	arvolume
	openff.evaluator.back	kends.dask), 213		1. (class method), 105	、 、	
data_cl	ass() (openff.evaluat	tor.storage.query.Ba	seDataQ	de jauit	_simulation_schema()) 41	
	class method), 235				(openff.evaluator.proper	ties.SolvationF	reeEnergy
data_cl	ass() (openff.evaluat	tor.storage.query.Bas	seSimula	tionData(Sugue method), 119	<i>(</i> ·	1 1
	class method), 241			default.	_storage_query()	(in	module
data_cl	ass() (openff.evaluat	tor.storage.query.For	rceFieldQ	Query	openff.evaluator.layers.r		
	class method), 238			default.	_unit()(openff.evaluate	or.datasets.Phy	sicalProperty

class method), 94	property), 382
	ndencies (openff.evaluator.protocols.miscellaneous.DivideValue
class method), 101	property), 397
	${\tt mdencies} \ (open ff. evaluator. protocols. miscellaneous. Dummy Protocols. Mathematical and the second state of the seco$
class method), 108	property), 411
	ndencies (openff.evaluator.protocols.miscellaneous.FilterSubstanceB
class method), 112	property), 407
	indiancies (openff.evaluator.protocols.miscellaneous.MultiplyValue
class method), 115	property), 392
	mdencies (openff.evaluator.protocols.miscellaneous.SubtractValues
class method), 105	property), 387
	<pre>xdfinities(openff.evaluator.protocols.miscellaneous.WeightByMoleFr</pre>
class method), 122	property), 402
	<pre>rgdencies (openff.evaluator.protocols.openmm.OpenMMEnergyMinin</pre>
class method), 119	property), 417
	ndencies (openff.evaluator.protocols.openmm.OpenMMEvaluateEner property), 429
(openff.evaluator.properties.HostGuestBindingAffinity static method), 122 dependence	
Density (class in openff.evaluator.properties), 100	ndencies (openff.evaluator.protocols.openmm.OpenMMSimulation property), 423
	ndeinCipest(mtenff.evaluator.protocols.paprika.analysis.AnalyzeAPRPl
property), 300	property), 471
	Edensiaes (openff.evaluator.protocols.paprika.analysis.ComputeReference
property), 305	property), 481
	mdebleies (openff.evaluator.protocols.paprika.analysis.ComputeSymm
property), 294	property), 476
	fulsacids (openff.evaluator.protocols.paprika.coordinates.AddDumm
property), 289	property), 445
	hderReites (bpenff.evaluator.protocols.paprika.coordinates.PreparePu
property), 315	property), 435
dependencies (openff.evaluator.protocols.analysis.Computedpe	ndeMainestcopenff.evaluator.protocols.paprika.coordinates.PrepareRel
property), 310	property), 440
dependencies (openff.evaluator.protocols.analysis.Decorrelepe	Mancales (openff.evaluator.protocols.paprika.restraints.ApplyRestrain
property), 326	property), 466
dependencies (openff.evaluator.protocols.analysis.Decorrelaped	តែdentaios (openff.evaluator.protocols.paprika.restraints.GenerateAtta
property), 321	property), 450
	ndanta.B:(dpeo)[f.evaluator.protocols.paprika.restraints.GeneratePull
property), 332	property), 455
	kddfoirdinapenff.evaluator.protocols.paprika.restraints.GenerateRele
property), 344	property), 461
	ristingStes (typenff.evaluator.protocols.reweighting.BaseEvaluateEnerg
property), 337	property), 497
	nclemcies (openff.evaluator.protocols.reweighting.BaseMBARProtoco
property), 349	property), 503
	rdensises (openff.evaluator.protocols.reweighting.ConcatenateObserv
property), 359	property), 492
	nflexates (openff.evaluator.protocols.reweighting.ConcatenateTraject
property), 354	property), 487
	Solution es (openff.evaluator.protocols.reweighting.ReweightDielectric
property), 365	property), 514
property), 371	ndencies (openff.evaluator.protocols.reweighting.ReweightObservabl property), 508
	property), 508 mdencies(openff.evaluator.protocols.simulation.BaseEnergyMinimiso
property), 377	property), 519
	hdencies (openff.evaluator.protocols.simulation.BaseSimulation
acpendence co (openg).evanantor.protocous.miscenancous.Macpen	menteres (openij), evaluation protocols, sinialation. Dase sinialation

property), 52		10 17	attribute), 552	(m 1	N 1
	evaluator.protocols.storage.Un	ipackSenæbl		(openff.evaluator.serve	r.Batch
property), 53		VL.Dere tele 1	attribute), 91		On an MME a survey Minimized
property), 53	Eevaluator.protocols.yank.Base	IankPenavol	attribute), 417	ttor.protocols.openmm.	OpenmmEnergyminimis
	9 Eevaluator.protocols.yank.Liga	ndRacamah	· · ·	tor protocols openmin	OnenMME valuate Energi
property), 54	· · ·	unec quabi	attribute), 429	uor.proiocois.openinin.	OpenminLvalaaleLnergi
	evaluator.protocols.yank.Solva	utionY œndaPor	· · ·	tor.protocols.openmm.	OpenMMSimulation
property), 55			attribute), 423	I	- I
	ppenff.evaluator.workflow.Proto	ocol enabl	e_pbc (openff.evalua	tor.protocols.reweighti	ng.BaseEvaluateEnergie.
property), 26			attribute), 496		
	Eevaluator.work flow.Protocol G	roup enabl		tor.protocols.simulatio	n.BaseEnergyMinimisati
property), 26			attribute), 518		
DielectricConstant		<i>in</i> enabl		tor.protocols.simulatio	n.BaseSimulation
	tor.properties), 107	. A	attribute), 525		21 - D - 41
attribute), 29	nff.evaluator.protocols.analysi. °	s.Averæ gedø æ	method), 249	tor.storage.attributes.F	ilePath
· · ·	° nff.evaluator.protocols.analysi.	Comonto D	,.		
attribute), 30		5.COMpenna <u>r</u> o		storage.data.StoredFre	eEnergyData
, · ·	^ nff.evaluator.protocols.reweigh	ting.Reweig		siorage.aana.siorearre	eEnergyD and
attribute), 51				tor.storage.attributes.F	FilePath
DivideValue	(class	in	method), 249	0	
openff.evalua	tor.protocols.miscellaneous),	ensen	ble(openff.evaluator	r.protocols.openmm.Op	penMMSimulation
394			attribute), 423		
	ator.protocols.analysis.Average	Obser enlsle n		r.protocols.simulation.E	BaseSimulation
attribute), 29			attribute), 525	<i>.</i> .	
	tor.protocols.miscellaneous.D	ivideV äin teha		(class	in
attribute), 39		En the	openff.evaluator.p	-	·
docked_complex_coc	ator.protocols.coordinates.Buil		lpyOfVaporizatio		in
attribute), 34	-		ated_properties	nopernes), 114	
docked_ligand_coor		COUL		client.RequestResult	at-
_	ator.protocols.coordinates.Buil	dDockedCo		enenniequestics	
attribute), 34				(openff.evaluator.serve	r.Batch
doi (openff.evaluator.	datasets.MeasurementSource		attribute), 91		
tribute), 99				n openff.evaluator.clier	ıt), 76
	E. evaluator. for cefield. LigParGet				
property), 18				n openff.evaluator.serv	
DummyProtocol	(class		· •	nff.evaluator.substance	
x 00	tor.protocols.miscellaneous),	excep		uator.client.RequestRest	ult at-
409		OVCOR	tribute), 86	tor.layers.CalculationL	averDegult
E		excel	attribute), 190	uor.tayers.CatculationL	LayerKesuti
	openff.evaluator.protocols.rew	eichtinen Ben		utor server Batch attribu	ute) 92
attribute), 50				ator.workflow.Workflow	
	_ (openff.evaluator.protocols.rew				
attribute), 51		0 0	sMolarVolume	(class	in
effective_samples(openff.evaluator.protocols.rew	eighting.Rev	veigh pObffeevable tor.p	properties), 104	
attribute), 50		execu		or.protocols.analysis.Av	verageDielectricConstant
electrostatic_lamb			method), 300	.	
	ator.protocols.yank.SolvationY	ankPr oweoi		or.protocols.analysis.Av	verageFreeEnergies
attribute), 55			method), 305	annuta alla an aluai- A-	waraaa Obaamaa ki a
electrostatic_lamb			method), 294	or.protocols.analysis.Av	verugeObservable
(openff.evalu	ator.protocols.yank.SolvationY	ληκ ε γοιοςοί	memou), 294		

execute() (openff.evaluator.protocols.analysis.BaseAverageRebse	teD (openff.evaluator.protocols.paprika.coordinates.AddDummyAto
method), 289	method), 445
execute() (openff.evaluator.protocols.analysis.BaseDecornected	tel (b) (b) penff.evaluator.protocols.paprika.coordinates.PreparePullCo
method), 315	method), 435
execute() (openff.evaluator.protocols.analysis.ComputeDipukdu	tweenff.evaluator.protocols.paprika.coordinates.PrepareRelease
method), 310	method), 440
execute() (openff.evaluator.protocols.analysis.Decorrelate (Xecur method), 326	value (openff.evaluator.protocols.paprika.restraints.ApplyRestraints method), 466
execute() (openff.evaluator.protocols.analysis.Decorrelate	tory) (openff.evaluator.protocols.paprika.restraints.GenerateAttachR
method), 321	method), 450
<pre>execute() (openff.evaluator.protocols.coordinates.BuildCoexdau</pre>	ttesPackpeonIff.evaluator.protocols.paprika.restraints.GeneratePullRest
method), 332	method), 455
execute() (openff.evaluator.protocols.coordinates.BuildDated	be(dimpenff.evaluator.protocols.paprika.restraints.GenerateReleased
method), 344	method), 461
execute() (openff.evaluator.protocols.coordinates.SolvateTexisting	4.Stfld(toprenff.evaluator.protocols.reweighting.BaseEvaluateEnergies
method), 337	method), 497
execute() (openff.evaluator.protocols.forcefield.BaseBuildExstem method), 349	te() (openff.evaluator.protocols.reweighting.BaseMBARProtocol method), 503
<pre>execute() (openff.evaluator.protocols.forcefield.BuildLigParGena</pre>	Sps @m(openff.evaluator.protocols.reweighting.ConcatenateObservable
method), 359	method), 492
execute() (openff.evaluator.protocols.forcefield.BuildSmir eageSys	ter() (openff.evaluator.protocols.reweighting.ConcatenateTrajectorie
method), 354	method), 487
execute() (openff.evaluator.protocols.forcefield.BuildTLeapsesta	tre() (openff.evaluator.protocols.reweighting.ReweightDielectricCon
method), 365	method), 514
<pre>execute() (openff.evaluator.protocols.gradients.ZeroGradientscu method), 371</pre>	te() (openff.evaluator.protocols.reweighting.ReweightObservable method), 508
execute() (openff.evaluator.protocols.groups.Conditional@mequ	te() (openff.evaluator.protocols.simulation.BaseEnergyMinimisation
method), 377	method), 520
execute() (openff.evaluator.protocols.miscellaneous.AddVahasu	te() (openff.evaluator.protocols.simulation.BaseSimulation
method), 382	method), 527
execute() (openff.evaluator.protocols.miscellaneous.DivideKekua	te() (openff.evaluator.protocols.storage.UnpackStoredSimulationDa
method), 397	method), 532
execute() (openff.evaluator.protocols.miscellaneous.Dummex@con method), 412	be() (openff.evaluator.protocols.yank.BaseYankProtocol method), 539
<pre>execute() (openff.evaluator.protocols.miscellaneous.Filter&nbsta</pre>	<pre>tre&ByRplenff.evaluator.protocols.yank.LigandReceptorYankProtocol</pre>
method), 407	method), 546
execute() (openff.evaluator.protocols.miscellaneous.Multiplydal	tre() (openff.evaluator.protocols.yank.SolvationYankProtocol
method), 392	method), 554
execute() (openff.evaluator.protocols.miscellaneous.Subtrextect method), 387	265
execute() (openff.evaluator.protocols.miscellaneous.WeigherBydda method), 402	<i>method</i>), 266
execute() (openff.evaluator.protocols.openmm.OpenMMEnzegyM method), 417	method), 270
execute() (openff.evaluator.protocols.openmm.OpenMMEexlevant method), 429	method), 257
execute() (openff.evaluator.protocols.openmm.OpenMMSiemeland	(<i>openff.evaluator.workflow.WorkflowGraph</i>
method), 423	<i>method</i>), 258
execute() (openff.evaluator.protocols.paprika.analysis.Analyses method), 471	BRAUES (openff.evaluator.storage.attributes.FilePath method), 249
execute() (openff.evaluator.protocols.paprika.analysis.Compred	tefdr.complone nts
method), 481	(openff.evaluator.protocols.miscellaneous.FilterSubstanceByRole
execute() (openff.evaluator.protocols.paprika.analysis.Computes method), 476	Sym atetibyGo);rΧ on

F	openff.evaluator.datasets.curation.components.filtering),
fidelity (openff.evaluator.datasets.CalculationSource	160 De De comi e Cohema de la cohema
attribute), 98 Filter FilePath (class in openff.evaluator.storage.attributes),	ByRacemicSchema (class in openff.evaluator.datasets.curation.components.filtering),
247	160
FilterByCharged (class in Filter	BySmiles (class in
openff.evaluator.datasets.curation.components.filtering), 164	openff.evaluator.datasets.curation.components.filtering), 166
	BySmilesSchema (class in
openff.evaluator.datasets.curation.components.filtering), 164	openff.evaluator.datasets.curation.components.filtering), 166
FilterByElements (class in Filter	
openff.evaluator.datasets.curation.components.filtering), 161	openff.evaluator.datasets.curation.components.filtering), 167
	BySmirksSchema (class in
openff.evaluator.datasets.curation.components.filtering), 161	openff.evaluator.datasets.curation.components.filtering), 167
	ByStereochemistry (class in
openff.evaluator.datasets.curation.components.filtering), 171	openff.evaluator.datasets.curation.components.filtering), 163
	ByStereochemistrySchema (class in
openff.evaluator.datasets.curation.components.filtering), 171	openff.evaluator.datasets.curation.components.filtering), 163
	BySubstances (class in
openff.evaluator.datasets.curation.components.filtering), 165	openff.evaluator.datasets.curation.components.filtering), 170
	BySubstancesSchema (class in
openff.evaluator.datasets.curation.components.filtering), 165	openff.evaluator.datasets.curation.components.filtering), 169
	ByTemperature (class in
openff.evaluator.datasets.curation.components.filtering), 159	openff.evaluator.datasets.curation.components.filtering), 157
	ByTemperatureSchema (class in
openff.evaluator.datasets.curation.components.filtering), 159	openff.evaluator.datasets.curation.components.filtering), 157
, i	Duplicates (class in
openff.evaluator.datasets.curation.components.filtering), 169	156
	DuplicatesSchema (class in
openff.evaluator.datasets.curation.components.filtering), 168	openff.evaluator.datasets.curation.components.filtering), 156
	ed_substance (openff.evaluator.protocols.miscellaneous.FilterSub
openff.evaluator.datasets.curation.components.filtering), 158 Filter	attribute), 406 SubstanceByRole (class in
FilterByPressureSchema (class in	openff.evaluator.protocols.miscellaneous), 404
openff.evaluator.datasets.curation.components.filtering), 158 final_	value_source (openff.evaluator.workflow.schemas.WorkflowSchem
FilterByPropertyTypes (class in	attribute), 279
openff.evaluator.datasets.curation.components.filtering],- 162	property), 255
FilterByPropertyTypesSchema (class in find()	
openff.evaluator.datasets.curation.components.filtering),	method), 250 PrintType (class in
102	openff.evaluator.datasets.curation.components.selection),
FilterByRacemic (class in	, w

179			attrib	ute), 546		
force_field_id (o)	penff.evaluator.server.Batch	at-	free_energy_	difference		
tribute), 91			(open	ff.evaluator.prote	ocols.yank.Solvatio	nYankProtocol
<pre>force_field_id(oper</pre>	nff.evaluator.storage.data.Bas	seSimu	ulationDataattrib	ute), 553		
attribute), 226	Ĵ		free_energy_	difference		
force_field_id(oper attribute), 233	nff.evaluator.storage.data.Sto	redFre		ff.evaluator.stora ute), 232	uge.data.StoredFree	eEnergyData
	uff.evaluator.storage.data.Sto	redSin			(class	in
attribute), 230)		openf	f.evaluator.storag	ge.query), 245	
<pre>force_field_id(oper</pre>	nff.evaluator.storage.query.Bo	aseSim	udation_DompQne	nts() (openff.ev	valuator.substances	.Substance
attribute), 241			class	method), 126		
force_field_id(open attribute), 245	nff.evaluator.storage.query.F1 5	reeEne		ject()(openff.e method),235	evaluator.storage.qu	uery.BaseDataQuery
force_field_id(oper attribute), 243	nff.evaluator.storage.query.Si 3	mulati		<pre>ject() (openff.e method), 241</pre>	evaluator.storage.qu	uery.BaseSimulationDat
<pre>force_field_path(op attribute), 348</pre>	penff.evaluator.protocols.forc	efield.		vject()(openff.e method),238	evaluator.storage.qu	uery.ForceFieldQuery
	penff.evaluator.protocols.forc	efield.		System) (openff.e method), 246	evaluator.storage.q	uery.FreeEnergyDataQu
, · ·	penff.evaluator.protocols.forc	efield.	Bfacksts_ndianaff_Sho	· · ·	evaluator.storage.qu	uery.SimulationDataQue
, · ·	penff.evaluator.protocols.forc	efield.	B£nilohTLdzorip&ys(te	, · ·	atasets.thermoml.T	ThermoMLDataSet
<pre>force_field_path(op</pre>	penff.evaluator.protocols.gra	dients.	Z eroc izedicepti	on() (<i>openff.eva</i>	luator.utils.excepti	ons.EvaluatorException
attribute), 369				method), 87	1 . 10 11	
attribute), 531		-	class	method), 257		
-	penff.evaluator.protocols.yan	k.Liga	-		datasets.thermoml.	.ThermoMLDataSet
attribute), 544		_		<i>method</i>), 143		~1
force_field_source attribute), 223	(openff.evaluator.storage.dat 3	ta.Forc		(openff.evaluator. method), 565	attributes.Attribut	eClass
force_field_source attribute), 238	(openff.evaluator.storage.que	ery.Fo		openff.evaluator. method), 81	client.Connection(Options
ForceFieldData	(class	in	<pre>from_json()</pre>	(openff.evaluat	tor.client.Request	class
openff.evaluat	tor.storage.data), 222		metho	od), 83		
ForceFieldQuery	(class	in	<pre>from_json()</pre>	(openff.evaluat	tor.client.Request0	ptions
1 00	tor.storage.query), 238			method), 85		
ForceFieldSource		in			ator.client.Request	tResult
	tor.forcefield), 181			method), 87		~
<i>method</i>), 250	uluator.storage.attributes.File		class.	method), 98		
<pre>format_map() (openff.</pre>	evaluator.storage.attributes.l	FilePat		openff.evaluator. method), 99	datasets.Measuren	ientSource
<pre>frame_counts(openff.</pre>	evaluator.protocols.reweight	ing.Ba		bpenff.evaluator. method), 95	datasets.PhysicalP	Property
	evaluator.protocols.reweight	ing.Re	w £ighnDjsbra (r)d		datasets.PhysicalP	PropertyDataSet
	evaluator.protocols.reweight	ing.Re	w £ighuOjson u@bl	· · ·	or.datasets.Source	class
free_energy_differ					datasets.taproom.T	Caproom Data Set
	ttor.protocols.yank.BaseYank	Protoc		method), 148	unusers.uproom.1	aproomDuiusei
attribute), 538		- 10100		· · ·	datasets.taproom.T	FaproomSource
free_energy_differ				method), 151		T
	utor.protocols.yank.LigandRe	ceptor		· · ·	datasets.thermoml.	.ThermoMLDataSet

class method), 143	class method), 338
<pre>from_json() (openff.evaluator.forcefield.ForceFieldSourcefrom_js</pre>	
class method), 181	class method), 349
<pre>from_json() (openff.evaluator.forcefield.LigParGenForceFirebatio_jus</pre>	
class method), 186	class method), 360
<pre>from_json() (openff.evaluator.forcefield.SmirnoffForceFielfdSomrjes</pre>	class method), 354
<pre>from_json() (openff.evaluator.forcefield.TLeapForceFieldScoremejs</pre>	on() (openff.evaluator.protocols.forcefield.BuildTLeapSystem class method), 366
from_json() (openff.evaluator.layers.CalculationLayerResfittom_js	
class method), 190	class method), 371
<pre>from_json() (openff.evaluator.layers.CalculationLayerSch&mom_js</pre>	
class method), 192	class method), 378
<pre>from_j son() (openff.evaluator.layers.reweighting.Reweightingticligs</pre>	oa() (openff.evaluator.protocols.groups.ConditionalGroup.Condit
class method), 202	class method), 375
from_json() (openff.evaluator.layers.simulation.Simulationfschnenja	
class method), 198	class method), 382
<pre>from_json() (openff.evaluator.layers.workflow.WorkflowCdframings class method), 195</pre>	Cheven (Construction: Construction: Constr
<pre>from_json() (openff.evaluator.properties.Density class from_js</pre>	on() (openff.evaluator.protocols.miscellaneous.DummyProtocol class method), 412
<pre>from_json() (openff.evaluator.properties.DielectricConstation_js</pre>	
class method), 109	class method), 407
<pre>from_json() (openff.evaluator.properties.EnthalpyOfMixinfgrom_js</pre>	on() (openff.evaluator.protocols.miscellaneous.MultiplyValue
class method), 113	class method), 392
<pre>from_json() (openff.evaluator.properties.EnthalpyOfVapoficoniojs</pre>	on() (openff.evaluator.protocols.miscellaneous.SubtractValues class method), 387
<pre>from_json() (openff.evaluator.properties.ExcessMolarVolutimem_js</pre>	
class method), 106	class method), 402
from_json() (openff.evaluator.properties.HostGuestBindingAffinijs	
class method), 123	class method), 417
<pre>from_json() (openff.evaluator.properties.SolvationFreeEn&rgom_js</pre>	
class method), 119	class method), 429
<pre>from_json() (openff.evaluator.protocols.analysis.AverageIfrebractrijs</pre>	class method), 423
from_j son() (openff.evaluator.protocols.analysis.AverageF fred ingi	
class method), 306	class method), 472
<pre>from_j son() (openff.evaluator.protocols.analysis.AverageOffrom_jds</pre>	
<i>class method</i>), 294	class method), 482
from_json() (openff.evaluator.protocols.analysis.BaseAverfigenOLijs	
class method), 289	class method), 477
<pre>from_json() (openff.evaluator.protocols.analysis.BaseDec&rantaijs class method), 316</pre>	bru@:(b penff.evaluator.protocols.paprika.coordinates.AddDummy/ class method), 445
from_j son() (openff.evaluator.protocols.analysis.Compute Dipmlej	
class method), 311	class method), 435
from_json() (openff.evaluator.protocols.analysis.Decorrelfite@ubsjes	
class method), 326	class method), 440
from_json() (openff.evaluator.protocols.analysis.Decorrelfitedinajes	condy (open ff.evaluator.protocols.paprika.restraints.Apply Restraints) (open ff.evaluator.protocols.paprika.restraints) (open ff.evaluator.paprika.restraints) (open ff.evalu
class method), 321	class method), 466
from_json() (openff.evaluator.protocols.coordinates.Buildfromdijs	
class method), 332	class method), 450
<pre>from_json() (openff.evaluator.protocols.coordinates.BuildErorked6</pre>	
class method), 344	class method), 456
<pre>from_json() (openff.evaluator.protocols.coordinates.Solvafafonistjs</pre>	gui u u umpeniji.evaluator.protocois.paprika.restraints.GenerateRelea

class method), 461	<i>method</i>), 131
<pre>from_json() (openff.evaluator.protocols.reweighting.BaseErrohmaj</pre>	
class method), 497	class method), 130
<pre>from_json() (openff.evaluator.protocols.reweighting.BaseMADMRff</pre>	
class method), 503	class method), 133
<pre>from_json() (openff.evaluator.protocols.reweighting.Concfitemutej</pre>	
class method), 492	class method), 135
<pre>from_json() (openff.evaluator.protocols.reweighting.Conc fiteom_tg)</pre>	
class method), 487	class method), 128
from_json() (openff.evaluator.protocols.reweighting.RewefglanDig	
class method), 514	class method), 137
from_json() (openff.evaluator.protocols.reweighting.Rewefgland)	
class method), 508	class method), 87
<pre>from_j son() (openff.evaluator.protocols.simulation.BaseEfirergy_M</pre>	
class method), 520	method), 265
from_json() (openff.evaluator.protocols.simulation.BaseSiEnnehatij	
class method), 527	class method), 270
<pre>from_json() (openff.evaluator.protocols.storage.UnpackStdmodfSinja class method), 533</pre>	sign(Onlopenff.evaluator.workflow.schemas.ProtocolGroupSchema class method), 275
<pre>from_json() (openff.evaluator.protocols.yank.BaseYankPr&momLj</pre>	<pre>son() (openff.evaluator.workflow.schemas.ProtocolReplicator</pre>
class method), 539	class method), 277
<pre>from_json() (openff.evaluator.protocols.yank.LigandRecepfrookarij</pre>	kscnv()dapenff.evaluator.workflow.schemas.ProtocolSchema
class method), 546	class method), 273
<pre>from_json() (openff.evaluator.protocols.yank.SolvationYanfkRomotg</pre>	codn() (openff.evaluator.workflow.schemas.WorkflowSchema
class method), 554	class method), 280
<pre>from_json() (openff.evaluator.server.Batch class from_j method), 92</pre>	<pre>son() (openff.evaluator.workflow.WorkflowException</pre>
<pre>from_json() (openff.evaluator.storage.data.BaseSimulationEponta_j</pre>	
class method), 227	class method), 260
<pre>from_json() (openff.evaluator.storage.data.BaseStoredDatfrom_o</pre>	
class method), 220	class method), 182
<pre>from_json() (openff.evaluator.storage.data.ForceFieldDatterom_o</pre>	penmm() (openff.evaluator.utils.observables.ObservableFrame
class method), 223	class method), 569
<pre>from_json() (openff.evaluator.storage.data.HashableStoreffDamap</pre>	andas() (openff.evaluator.datasets.PhysicalPropertyDataSet
class method), 221	class method), 141
<pre>from_json() (openff.evaluator.storage.data.ReplaceableDdfmom_p</pre>	andas() (openff.evaluator.datasets.taproom.TaproomDataSet
class method), 224	class method), 148
<pre>from_json() (openff.evaluator.storage.data.StoredFreeEnefyphap</pre>	mandas() (openff.evaluator.datasets.thermoml.ThermoMLDataSet
class method), 233	class method), 143
<pre>from_json() (openff.evaluator.storage.data.StoredSimulationDutp</pre>	ath() (openff.evaluator.forcefield.SmirnoffForceFieldSource
class method), 230	class method), 182
<pre>from_json() (openff.evaluator.storage.query.BaseDataQuefryom_s</pre>	<pre>chema() (openff.evaluator.protocols.analysis.AverageDielectricCon class method), 300</pre>
from_j son() (openff.evaluator.storage.query.BaseSimulationDates	
class method), 241	class method), 306
<pre>from_json() (openff.evaluator.storage.query.ForceFieldQufergom_s</pre>	
z = z class method), 239	class method), 294
from_json() (openff.evaluator.storage.query.FreeEnergyDfittofues	
class method), 246	class method), 289
from_json() (openff.evaluator.storage.query.SimulationDataQmes	
class method), 243	class method), 316
<pre>from_json() (openff.evaluator.storage.query.SubstanceQueryom_s</pre>	
class method), 237	class method), 311
<pre>from_json() (openff.evaluator.substances.Amount class from_s</pre>	

	class method), 467
<pre>from_schema() (openff.evaluator.protocols.analysis.DecorfademeTsafe</pre>	xentar(y) (openff.evaluator.protocols.paprika.restraints.GenerateAtt
class method), 321	class method), 451
from_schema() (openff.evaluator.protocols.coordinates.Bufid@w_sdh	uentesP a(d upen)ff.evaluator.protocols.paprika.restraints.GeneratePu
class method), 332	class method), 456
from_schema() (openff.evaluator.protocols.coordinates.Bufidom_set	lena (dihapenff.evaluator.protocols.paprika.restraints.GenerateRe
class method), 344	class method), 461
from_schema() (openff.evaluator.protocols.coordinates.SolfmonE_siste	ivegst ()) (toppenff.evaluator.protocols.reweighting.BaseEvaluateEner
class method), 338	class method), 497
from_schema() (openff.evaluator.protocols.forcefield.Base BuddSsoft	wema() (openff.evaluator.protocols.reweighting.BaseMBARProtoc
class method), 349	class method), 503
from_schema() (openff.evaluator.protocols.forcefield.BuildfrgRasGe	weight (openff.evaluator.protocols.reweighting.ConcatenateObser
	class method), 492
from_schema() (openff.evaluator.protocols.forcefield.BuildSmannsffS	isstan) (openff.evaluator.protocols.reweighting.ConcatenateTrajed
	class method), 487
from_schema() (openff.evaluator.protocols.forcefield.BuildfffampSgk	
	class method), 514
from_schema() (openff.evaluator.protocols.gradients.ZeroGradiestch	
	class method), 508
from_schema() (openff.evaluator.protocols.groups.Conditionalg.sat	
	class method), 520
from_schema() (openff.evaluator.protocols.miscellaneous.AudMabach	
	class method), 527
from_schema() (openff.evaluator.protocols.miscellaneous. Dronde Kah	
	class method), 533
from_schema() (openff.evaluator.protocols.miscellaneous.Dnomn_sPr	
	class method), 539
from_schema() (openff.evaluator.protocols.miscellaneous. Filtem_Sabk	
	class method), 546
from_schema() (openff.evaluator.protocols.miscellaneous. Mindnip 50%	
	class method), 555
from_schema() (openff.evaluator.protocols.miscellaneous.Submast	
	class method), 262
from_schema() (openff.evaluator.protocols.miscellaneous. Weights	
	class method), 270
from_schema() (openff.evaluator.protocols.openmm.OpenMfMoffinesch	
	class method), 256
from_schema() (openff.evaluator.protocols.openmm.OpenNffMoffivestur	
	class method), 96
from_schema() (openff.evaluator.protocols.openmm.Open/ffn/6fin/mark	
	class method), 143
from_schema() (openff.evaluator.protocols.paprika.analysiscomalyme	
	class method), 144
from_schema() (openff.evaluator.protocols.paprika.analysisuCompart	
	property), 285
from_schema() (openff.evaluator.protocols.paprika.analysifsuCamput	
	attribute), 400
from_schema() (openff.evaluator.protocols.paprika.coordinates.Add	DummyAtoms
class method), 446 G	
from_schema() (openff.evaluator.protocols.paprika.coordinates.Pare	
class method), 435	ule openff.evaluator.protocols.utils), 561
from_schema() (openff.evaluator.protocols.paprika.coordinates.Pare	
class method), 440	(openff.evaluator.workflow.Workflow static
<pre>from_schema() (openff.evaluator.protocols.paprika.restraints.Apply]</pre>	KAEKKHATJ, S256

<pre>generate_reweighting_protocols() (in module</pre>	class method), 300
openff.evaluator.protocols.utils), 562	get_attributes() (openff.evaluator.protocols.analysis.AverageFreeEnerg
<pre>generate_simulation_protocols() (in module</pre>	class method), 306
openff.evaluator.protocols.utils), 562 GenerateAttachRestraints (class in	<pre>get_attributes() (openff.evaluator.protocols.analysis.AverageObservab</pre>
GenerateAttachRestraints (class in openff.evaluator.protocols.paprika.restraints),	get_attributes() (openff.evaluator.protocols.analysis.BaseAverageObse
448	class method), 289
GeneratePullRestraints (class in	get_attributes() (openff.evaluator.protocols.analysis.BaseDecorrelateF
openff.evaluator.protocols.paprika.restraints),	class method), 316
453	<pre>get_attributes() (openff.evaluator.protocols.analysis.ComputeDipoleM</pre>
GenerateReleaseRestraints (class in	class method), 311
openff.evaluator.protocols.paprika.restraints), 459	<pre>get_attributes() (openff.evaluator.protocols.analysis.DecorrelateObset</pre>
get() (openff.evaluator.utils.observables.ObservableFrame	eget_attributes() (openff.evaluator.protocols.analysis.DecorrelateTrajec
<i>method</i>), 570	class method), 321
	get_attributes() (openff.evaluator.protocols.coordinates.BuildCoordinates.
method), 127	class method), 332
get_attributes() (openff.evaluator.attributes.AttributeC class method), 565	<pre>'kgest_attributes() (openff.evaluator.protocols.coordinates.BuildDockedC class method), 344</pre>
	tions_attributes() (openff.evaluator.protocols.coordinates.SolvateExistin
class method), 81	class method), 338
<pre>get_attributes() (openff.evaluator.client.Request</pre>	get_attributes() (openff.evaluator.protocols.forcefield.BaseBuildSystem
class method), 83	class method), 350
<pre>get_attributes() (openff.evaluator.client.RequestOption</pre>	<pre>uget_attributes() (openff.evaluator.protocols.forcefield.BuildLigParGen</pre>
	get_attributes() (openff.evaluator.protocols.forcefield.BuildSmirnoffSy.
class method), 87	class method), 354
<pre>get_attributes() (openff.evaluator.datasets.PhysicalPro</pre>	pgettyattributes() (openff.evaluator.protocols.forcefield.BuildTLeapSyste
class method), 95	class method), 366
<pre>get_attributes() (openff.evaluator.layers.CalculationLa</pre>	getRattributes() (openff.evaluator.protocols.gradients.ZeroGradients class method), 371
get_attributes() (openff.evaluator.layers.CalculationLa	getSchennibutes() (openff.evaluator.protocols.groups.ConditionalGroup
class method), 192	class method), 378
	geetighting Scheenes () (openff.evaluator.protocols.groups.ConditionalGroup.
class method), 202	class method), 375
class method), 198	rgdationSationates() (openff.evaluator.protocols.miscellaneous.AddValues class method), 383
	ciass memoa), 565 Gev Cal chimes the nupenff.evaluator.protocols.miscellaneous.DivideValue
class method), 195	class method), 397
	get_attributes() (openff.evaluator.protocols.miscellaneous.DummyProt
class method), 102	class method), 412
	Genstantributes() (openff.evaluator.protocols.miscellaneous.FilterSubsta
class method), 109	class method), 407
	Openflow
class method), 113	class method), 392
	分をなりませい () (openff.evaluator.protocols.miscellaneous.SubtractVal
class method), 116	class method), 387
	get/otume ibutes() (openff.evaluator.protocols.miscellaneous.WeightByM
class method), 106	class method), 402
class method), 123	<pre>tBitdiagt4ffibityes() (openff.evaluator.protocols.openmm.OpenMMEnergyl class method), 417</pre>
	<pre>GetEatigyibutes() (openff.evaluator.protocols.openmm.OpenMMEvaluat</pre>
class method), 119	class method), 429
	vgetgeDielexbuit Eshyt(menff.evaluator.protocols.openmm.OpenMMSimulat

class method), 423	class method), 221
get_attributes()(openff.evaluator.protocols.paprika.anglestisa4n	
class method), 472	class method), 224
get_attributes()(<i>openff.evaluator.protocols.paprika.ang</i> b <u>tis</u> a@t	
class method), 482	class method), 233
<pre>get_attributes() (openff.evaluator.protocols.paprika.angbtisaCto</pre>	<i>uppowerSystem (metry Configuration and class method)</i> , 230
get_attributes() (openff.evaluator.protocols.paprika.cogetinates	xidalDars hdy(apemff.evaluator.storage.query.BaseDataQuery
class method), 446	class method), 235
get_attributes() (openff.evaluator.protocols.paprika.cogetinates	HaptuesKi)llGpemffinutduator.storage.query.BaseSimulationDataQ
class method), 435	class method), 241
get_attributes() (openff.evaluator.protocols.paprika.cogetinates	AliopticsKeleopeoffandihuttar.storage.query.ForceFieldQuery
class method), 440	class method), 239
get_attributes() (openff.evaluator.protocols.paprika.resganinatA	pplyReas(intopenff.evaluator.storage.query.FreeEnergyDataQuer
class method), 467	class method), 246
get_attributes() (openff.evaluator.protocols.paprika.resgeninatG	
class method), 451	class method), 243
get_attributes() (openff.evaluator.protocols.paprika.resgentinestG	
class method), 456	class method), 237
get_attributes() (<i>openff.evaluator.protocols.paprika.res</i> getinestG	
class method), 461	class method), 132
get_attributes()(<i>openff.evaluator.protocols.reweightingBusa</i> Et	
class method), 498	class method), 130
get_attributes()(openff.evaluator.protocols.reweighting)BusaMt	
class method), 503	class method), 133
get_attributes() (openff.evaluator.protocols.reweighting Conatute	
class method), 492	class method), 135
get_attributes() (openff.evaluator.protocols.reweighting) Conatute	
class method), 487	class method), 128
get_attributes() (openff.evaluator.protocols.reweighting	
class method), 514	class method), 137
get_attributes() (openff.evaluator.protocols.reweightingftewatg	class method), 265
class method), 508	
get_attributes() (openff.evaluator.protocols.simulation.get_Ente class method), 520	class method), 270
get_attributes() (openff.evaluator.protocols.simulation.genseSiver	
class method), 527	class method), 275
get_attributes() (openff.evaluator.protocols.storage.Un ge ckSatura	
class method), 533	class method), 273
get_attributes() (openff.evaluator.protocols.yank.Base YgerkPawt	
class method), 539	class method), 280
get_attributes() (openff.evaluator.protocols.yank.LigangRecept	
class method), 546	class method), 260
get_attributes() (openff.evaluator.protocols.yank.SolvatjetYolka	
class method), 555	(openff.evaluator.protocols.analysis.AverageDielectricConstant
<pre>get_attributes() (openff.evaluator.server.Batch class</pre>	method), 300
	.ss_attribute()
get_attributes() (openff.evaluator.storage.data.BaseSimulationD	
class method), 227	<i>method</i>), 306
get_attributes() (openff.evaluator.storage.data.BaseStogetDatha	
class method), 220	(openff.evaluator.protocols.analysis.AverageObservable
<pre>get_attributes() (openff.evaluator.storage.data.ForceFieldData</pre>	
	ss_attribute()
get_attributes() (openff.evaluator.storage.data.HashableStoredL	

<i>method</i>), 289		<i>method</i>), 392
<pre>get_class_attribute()</pre>	-	ss_attribute()
$(open {\it ff.evaluator.protocols.analysis.BaseDecorrection)$	elateProtoc	${\it Q} {\it b} {\it penff.evaluator.protocols.miscellaneous.SubtractValues}$
<i>method</i>), 316		<i>method</i>), 388
get_class_attribute()	-	ss_attribute()
(openff.evaluator.protocols.analysis.ComputeDip		${\it l}\!\infty {\it penff.evaluator.protocols.miscellaneous.Weight By Mole Fraction}$
<i>method</i>), 311		<i>method</i>), 402
get_class_attribute()	-	ss_attribute()
		& open ff. evaluator. protocols. open mm. Open MME nergy Minimisation and the set of t
method), 326		method), 417
<pre>get_class_attribute()</pre>		ss_attribute()
		(openff.evaluator.protocols.openmm.OpenMMEvaluateEnergies
<i>method</i>), 321		method), 429
<pre>get_class_attribute()</pre>	-	ss_attribute()
(openff.evaluator.protocols.coordinates.BuildCoc method), 333		(chpeon/ff.evaluator.protocols.openmm.OpenMMSimulation method), 423
<pre>get_class_attribute()</pre>		ss_attribute()
		inapesnff.evaluator.protocols.paprika.analysis.AnalyzeAPRPhase
method), 345		method), 472
<pre>get_class_attribute()</pre>	get_cla	ss_attribute()
(openff.evaluator.protocols.coordinates.SolvateE	xistingStru	(http://www.computeReferencew) (http://www.computeRef
<i>method</i>), 338		<i>method</i>), 482
<pre>get_class_attribute()</pre>	get_cla	ss_attribute()
(openff.evaluator.protocols.forcefield.BaseBuildS	ystem	$(open {\it ff.evaluator.protocols.paprika.analysis.Compute Symmetry Comparison of the state of t$
<i>method</i>), 350		<i>method</i>), 477
<pre>get_class_attribute()</pre>		ss_attribute()
(openff.evaluator.protocols.forcefield.BuildLigPa method), 360		n(openff.evaluator.protocols.paprika.coordinates.AddDummyAtom method), 446
<pre>get_class_attribute()</pre>		ss_attribute()
-	-	(openff.evaluator.protocols.paprika.coordinates.PreparePullCoor
method), 355		method), 435
<pre>get_class_attribute()</pre>		ss_attribute()
(openff.evaluator.protocols.forcefield.BuildTLeap	-	(openff.evaluator.protocols.paprika.coordinates.PrepareReleaseC
method), 366		<i>method</i>), 440
<pre>get_class_attribute()</pre>	get_cla	ss_attribute()
(openff.evaluator.protocols.gradients.ZeroGradie method), 371		(openff.evaluator.protocols.paprika.restraints.ApplyRestraints method), 467
<pre>get_class_attribute()</pre>		ss_attribute()
(openff.evaluator.protocols.groups.ConditionalG		(openff.evaluator.protocols.paprika.restraints.GenerateAttachRest
method), 378	-	method), 451
<pre>get_class_attribute()</pre>		ss_attribute()
(openff.evaluator.protocols.miscellaneous.AddVa	-	(openff.evaluator.protocols.paprika.restraints.GeneratePullRestra
method), 383		method), 456
<pre>get_class_attribute()</pre>		ss_attribute()
(openff.evaluator.protocols.miscellaneous.Divide	Value	(openff.evaluator.protocols.paprika.restraints.GenerateReleaseRe
method), 397		method), 461
<pre>get_class_attribute()</pre>	get_cla	ss_attribute()
(openff.evaluator.protocols.miscellaneous.Dumm	yProtocol	(openff.evaluator.protocols.reweighting.BaseEvaluateEnergies
<i>method</i>), 412		<i>method</i>), 498
<pre>get_class_attribute()</pre>		ss_attribute()
(openff.evaluator.protocols.miscellaneous.FilterSilt	ubstanceB	(Rplanff.evaluator.protocols.reweighting.BaseMBARProtocol
<pre>method), 407 get_class_attribute()</pre>		<pre>method), 503 ss_attribute()</pre>
-	-	(openff.evaluator.protocols.reweighting.ConcatenateObservables
(open), $evalualor, prolocols, miscellaneous, Multip$	іучание	(openij.evalualor.protocols.reweighting.ConcalenaleObservables

<i>method</i>), 492	method), 333
<pre>get_class_attribute()</pre>	get_value() (openff.evaluator.protocols.coordinates.BuildDockedCoordin
(open ff. evaluator. protocols. reweighting. Concatered and the constant of	nateTrajectoniethod), 345
<i>method</i>), 488	get_value() (openff.evaluator.protocols.coordinates.SolvateExistingStruct
get_class_attribute()	method), 338
(openff.evaluator.protocols.reweighting.Reweight method), 514	t Djete_val:Gen}t(op enff.evaluator.protocols.forcefield.BaseBuildSystem method), 350
get_class_attribute()	get_value() (openff.evaluator.protocols.forcefield.BuildLigParGenSystem
(openff.evaluator.protocols.reweighting.Reweighting)	
method), 509	<pre>get_value() (openff.evaluator.protocols.forcefield.BuildSmirnoffSystem</pre>
<pre>get_class_attribute()</pre>	method), 355
	gglftniwaisaute(n) (openff.evaluator.protocols.forcefield.BuildTLeapSystem
method), 520	method), 366
<pre>get_class_attribute()</pre>	get_value() (openff.evaluator.protocols.gradients.ZeroGradients
(openff.evaluator.protocols.simulation.BaseSimu	
method), 527	get_value() (openff.evaluator.protocols.groups.ConditionalGroup
<pre>get_class_attribute()</pre>	method), 378
method), 533	ใ Sjetu_lมนี่ชนฺ฿()) d(openff.evaluator.protocols.miscellaneous.AddValues method), 383
<pre>get_class_attribute()</pre>	get_value() (openff.evaluator.protocols.miscellaneous.DivideValue
(openff.evaluator.protocols.yank.BaseYankProtoc	
method), 539	get_value() (openff.evaluator.protocols.miscellaneous.DummyProtocol
<pre>get_class_attribute()</pre>	method), 412
method), 546	Ygat Pwabael() (openff.evaluator.protocols.miscellaneous.FilterSubstanceBy method), 408
<pre>get_class_attribute()</pre>	get_value() (openff.evaluator.protocols.miscellaneous.MultiplyValue rotocol method), 392
method), 555	get_value() (openff.evaluator.protocols.miscellaneous.SubtractValues
<pre>get_class_attribute()</pre>	<i>method</i>), 388
(openff.evaluator.workflow.Protocol method), 264	<pre>get_value() (openff.evaluator.protocols.miscellaneous.WeightByMoleFra method), 402</pre>
<pre>get_class_attribute()</pre>	get_value() (openff.evaluator.protocols.openmm.OpenMMEnergyMinimi
(openff.evaluator.workflow.ProtocolGroup	method), 417
method), 269	get_value() (openff.evaluator.protocols.openmm.OpenMMEvaluateEnerg
<pre>get_molecules_per_component()</pre>	method), 429
(openff.evaluator.substances.Substance	get_value() (openff.evaluator.protocols.openmm.OpenMMSimulation
<i>method</i>), 127	<i>method</i>), 424
get_value() (openff.evaluator.protocols.analysis.Average	Djetecval: Go(s)t(mtenff.evaluator.protocols.paprika.analysis.AnalyzeAPRPha
<i>method</i>), 300	<i>method</i>), 472
get_value() (openff.evaluator.protocols.analysis.Average	FgetEnedgie() (openff.evaluator.protocols.paprika.analysis.ComputeReference
method), 306	<i>method</i>), 482
get_value() (openff.evaluator.protocols.analysis.Average	Qese_value() (openff.evaluator.protocols.paprika.analysis.ComputeSymme
method), 295	<i>method</i>), 477
	e rgge_QbskueD}(openff.evaluator.protocols.paprika.coordinates.AddDummy
method), 289	<i>method</i>), 446
<pre>get_value() (openff.evaluator.protocols.analysis.BaseDe</pre>	cgetelsta PuerOcobpenff.evaluator.protocols.paprika.coordinates.PreparePull method), 435
get_value() (openff.evaluator.protocols.analysis.Comput	te @pp_lealIue e())t(copenff.evaluator.protocols.paprika.coordinates.PrepareRele
<i>method</i>), 311	<i>method</i>), 440
get_value() (openff.evaluator.protocols.analysis.Decorre	el ge:Qbs&rue& @e \$ openff.evaluator.protocols.paprika.restraints.ApplyRestraint
<i>method</i>), 326	<i>method</i>), 467
	el gæf[wjækuæý) (openff.evaluator.protocols.paprika.restraints.GenerateAttac
<i>method</i>), 321	<i>method</i>), 451
get_value() (openff.evaluator.protocols.coordinates.Buil	d Gwo<u>r</u>dialatesPa(akpen/f f.evaluator.protocols.paprika.restraints.GeneratePullR

<i>method</i>), 456		<i>method</i>), 321
get_value() (openff.evaluator.protocols.paprika.restraints	gGe <u>en</u> smad	
<i>method</i>), 461		(open ff. evaluator. protocols. coordinates. Build Coordinates Packmone and the second seco
get_value() (openff.evaluator.protocols.reweighting.Based		
	-	ue_references()
<i>method</i>), 503		o(opehff.evaluator.protocols.coordinates.BuildDockedCoordinates method), 345
<pre>get_value() (openff.evaluator.protocols.reweighting.Conc</pre>	gæn <i>a</i> væØ	
method), 492		(openff.evaluator.protocols.coordinates.SolvateExistingStructure
get_value() (openff.evaluator.protocols.reweighting.Conc		•
<pre>method), 488 get_value() (openff.evaluator.protocols.reweighting.Rewe</pre>	-	ue_references()
method), 515	igniDiele	method), 350
<pre>get_value() (openff.evaluator.protocols.reweighting.Rewe</pre>	ind that	
method), 509	AGUE TOUL	(openff.evaluator.protocols.forcefield.BuildLigParGenSystem
get_value() (openff.evaluator.protocols.simulation.BaseE	nerøvMin	
-		ue_references()
get_value() (openff.evaluator.protocols.simulation.BaseSi	-	
method), 527		method), 355
get_value() (openff.evaluator.protocols.storage.UnpackSt	græt <i>l Svir</i> all	dur <u>imelfau</u> rences()
<i>method</i>), 533		(openff.evaluator.protocols.forcefield.BuildTLeapSystem
<pre>get_value() (openff.evaluator.protocols.yank.BaseYankPr</pre>	otocol	<i>method</i>), 366
		ue_references()
<pre>get_value() (openff.evaluator.protocols.yank.LigandRecep method), 547</pre>	ptorYank	Propenfl.evaluator.protocols.gradients.ZeroGradients method), 372
<pre>get_value() (openff.evaluator.protocols.yank.SolvationYan</pre>	glaftr <u>o</u> tad	nde_references()
<pre>method), 555 get_value() (openff.evaluator.workflow.Protocol</pre>		(openff.evaluator.protocols.groups.ConditionalGroup method), 376
	get_val	ue_references()
<pre>get_value() (openff.evaluator.workflow.ProtocolGroup</pre>		(openff.evaluator.protocols.miscellaneous.AddValues method), 383
	get_val	ue_references()
(openff.evaluator.protocols.analysis.AverageDieled method), 301	ctricCons	st (op enff.evaluator.protocols.miscellaneous.DivideValue method), 397
	get_val	ue_references()
	Energies	(openff.evaluator.protocols.miscellaneous.DummyProtocol method), 412
	get_val	ue_references()
(openff.evaluator.protocols.analysis.AverageObser method), 295		(openff.evaluator.protocols.miscellaneous.FilterSubstanceByRole method), 408
	get_val	ue_references()
		leopenff.evaluator.protocols.miscellaneous.MultiplyValue method), 393
	get_val	ue_references()
		c&penff.evaluator.protocols.miscellaneous.SubtractValues method), 388
	det val	ue_references()
-	-	<i>uc_references()</i> <i>ucopenff.evaluator.protocols.miscellaneous.WeightByMoleFraction</i>
method), 311		<i>method</i>), 402
	get_val	ue_references()
-	-	e(openff.evaluator.protocols.openmm.OpenMMEnergyMinimisatio method), 418
	get_val	ue_references()
-	-	(openff.evaluator.protocols.openmm.OpenMMEvaluateEnergies

<i>method</i>), 430	<i>method</i>), 520
<pre>get_value_references()</pre>	<pre>get_value_references()</pre>
(openff.evaluator.protocols.openmm.Op	
<i>method</i>), 424	<i>method</i>), 527
<pre>get_value_references()</pre>	<pre>get_value_references()</pre>
<i>method</i>), 472	alysis.AnalyzeAPRPh(uspenff.evaluator.protocols.storage.UnpackStoredSimulationData method), 533
<pre>get_value_references()</pre>	<pre>get_value_references()</pre>
(openff.evaluator.protocols.paprika.and method), 482	alysis.ComputeRefere(upe8//frk valuator.protocols.yank.BaseYankProtocol method), 540
<pre>get_value_references()</pre>	<pre>get_value_references()</pre>
	alysis.ComputeSymm &try&uff:eexciliuun tor.protocols.yank.LigandReceptorYankProtocol
method), 477	method), 547
<pre>get_value_references()</pre>	<pre>get_value_references()</pre>
method), 446	ordinates.AddDummy (copens ff.evaluator.protocols.yank.SolvationYankProtocol method), 555
<pre>get_value_references()</pre>	<pre>get_value_references()</pre>
method), 435	ordinates.PreparePul l6pemffinutdu ator.workflow.Protocol method), 263
<pre>get_value_references()</pre>	<pre>get_value_references()</pre>
	ordinates.PrepareRel (ope Gffandihuttasr.workflow.ProtocolGroup
method), 440	method), 269
<pre>get_value_references()</pre>	gpu_device_indices (openff.evaluator.backends.ComputeResources
(openff.evaluator.protocols.paprika.res	
<pre>method), 467 get_value_references()</pre>	<pre>gpu_device_indices(openff.evaluator.backends.QueueWorkerResources</pre>
(openff.evaluator.protocols.paprika.res	
method), 451	(openff.evaluator.protocols.analysis.ComputeDipoleMoments
<pre>get_value_references()</pre>	attribute), 309
(openff.evaluator.protocols.paprika.res	
method), 456	(openff.evaluator.protocols.gradients.ZeroGradients
<pre>get_value_references()</pre>	attribute), 370
(openff.evaluator.protocols.paprika.res	
<i>method</i>), 462	$(open f\!\!f. evaluator. protocols. open mm. Open MME valuate Energies$
<pre>get_value_references()</pre>	attribute), 430
(openff.evaluator.protocols.reweighting	
method), 498	(openff.evaluator.protocols.openmm.OpenMMSimulation
<pre>get_value_references()</pre>	attribute), 424
(openj).evaluator.protocols.reweighting method), 504	(openff.evaluator.protocols.reweighting.BaseEvaluateEnergies
<pre>get_value_references()</pre>	attribute), 496
(openff.evaluator.protocols.reweighting	
method), 492	(openff.evaluator.protocols.simulation.BaseSimulation
<pre>get_value_references()</pre>	attribute), 525
(openff.evaluator.protocols.reweighting	
<i>method</i>), 488	(openff.evaluator.protocols.yank.BaseYankProtocol
<pre>get_value_references()</pre>	attribute), 537
(openff.evaluator.protocols.reweighting	
<i>method</i>), 515	(open ff. evaluator. protocols. yank. Ligand Receptor Yank Protocol
<pre>get_value_references()</pre>	attribute), 547
(openff.evaluator.protocols.reweighting	
method), 509	(openff.evaluator.protocols.yank.SolvationYankProtocol attribute) 555
<pre>get_value_references()</pre>	attribute), 555 BaseEnerg yIndivinati on(openff.evaluator.datasets.PhysicalProperty

attribute), 95	high_precision (openff.evaluator.protocols.openmm.OpenMMSimulation
gradients (openff.evaluator.properties.Density at- tribute), 102	attribute), 424 high_precision(openff.evaluator.protocols.simulation.BaseSimulation
gradients (openff.evaluator.properties.DielectricConstant	
attribute), 109	host_coordinate_path
gradients (openff.evaluator.properties.EnthalpyOfMixing	-
attribute), 113	attribute), 460
gradients (openff.evaluator.properties.EnthalpyOfVaporiz	
attribute), 116	openff.evaluator.properties), 121
<pre>gradients(openff.evaluator.properties.ExcessMolarVolun</pre>	ime
gradients (openff.evaluator.properties.HostGuestBinding.	gAffinithenff.evaluator.client.Reauest attribute), 82
attribute), 123	id (openff.evaluator.datasets.PhysicalProperty attribute),
gradients (openff.evaluator.properties.SolvationFreeEner	ergy 94
attribute), 119	id (openff.evaluator.properties.Density attribute), 102
<pre>gradients (openff.evaluator.workflow.WorkflowResult</pre>	id (openff.evaluator.properties.DielectricConstant
attribute), 260	attribute), 109
guest_orientation_mask	id (openff.evaluator.properties.EnthalpyOfMixing
(openff.evaluator.protocols.paprika.coordinates.F	
attribute), 433	id (openff.evaluator.properties.EnthalpyOfVaporization
Н	attribute), 116
	id (openff.evaluator.properties.ExcessMolarVolume at-
has_ancillary_data()	tribute), 106
	atud (openff.evaluator.properties.HostGuestBindingAffinity
class method), 227	attribute), 123
has_ancillary_data()	id (openff.evaluator.properties.SolvationFreeEnergy at-
(openff.evaluator.storage.data.BaseStoredData class method), 219	tribute), 119 id(openff.evaluator.protocols.analysis.AverageDielectricConstant
has_ancillary_data()	attribute), 301
(openff.evaluator.storage.data.ForceFieldData	id (openff.evaluator.protocols.analysis.AverageFreeEnergies
class method), 223	attribute), 307
has_ancillary_data()	id (openff.evaluator.protocols.analysis.AverageObservable
(openff.evaluator.storage.data.HashableStoredData)	ata attribute), 295
class method), 221	$\verb"id" (open ff. evaluator. protocols. analysis. Base Average Observable$
has_ancillary_data()	attribute), 290
	$\verb"id" (open ff. evaluator. protocols. analysis. Base Decorrelate Protocol" (open ff. evaluator. protocols. analysis. Base Decorrelate Protocol")$
class method), 225	attribute), 317
has_ancillary_data()	id (openff.evaluator.protocols.analysis.ComputeDipoleMoments
(openff.evaluator.storage.data.StoredFreeEnergyl	
class method), 233	id (openff.evaluator.protocols.analysis.DecorrelateObservables
has_ancillary_data()	attribute), 327
(openjj.evaluator.storage.aata.storeasimulationD class method), 230	Data (openff.evaluator.protocols.analysis.DecorrelateTrajectory attribute) 322
	attribute), 322 St àduge penff.evaluator.protocols.coordinates.BuildCoordinatesPackmol
method), 217	attribute), 333
	all tollie), 555 ackedt(bpenff.evaluator.protocols.coordinates.BuildDockedCoordinates
method), 216	attribute), 345
	e id(openff.evaluator.protocols.coordinates.SolvateExistingStructure
method), 217	attribute), 339
	id (openff.evaluator.protocols.forcefield.BaseBuildSystem
<i>method</i>), 216	attribute), 350
HashableStoredData (class in	id(openff.evaluator.protocols.forcefield.BuildLigParGenSystem
openff.evaluator.storage.data), 221	attribute), 361

id (openff.evaluator.protocols.forcefield.BuildSmirnoffSystemid (open	ff.evaluator.protocols.reweighting.ConcatenateTrajectories attribute), 488
attribute), 355 id (openff.evaluator.protocols.forcefield.BuildTLeapSystem id (open	
attribute), 366	attribute), 515
id (openff.evaluator.protocols.gradients.ZeroGradients id(open attribute), 372	ff.evaluator.protocols.reweighting.ReweightObservable attribute), 509
id (openff.evaluator.protocols.groups.ConditionalGroup id(open attribute), 378	ff.evaluator.protocols.simulation.BaseEnergyMinimisation attribute), 521
id (openff.evaluator.protocols.miscellaneous.AddValues id(open attribute), 383	ff.evaluator.protocols.simulation.BaseSimulation attribute), 528
id (openff.evaluator.protocols.miscellaneous.DivideValue id (open attribute), 398	ff.evaluator.protocols.storage.UnpackStoredSimulationData attribute), 534
id (openff.evaluator.protocols.miscellaneous.DummyProtocad (open	ff.evaluator.protocols.yank.BaseYankProtocol at-
attribute), 413	tribute), 540
id (openff.evaluator.protocols.miscellaneous.FilterSubstancals(Rplan attribute), 408	attribute), 547
id (openff.evaluator.protocols.miscellaneous.MultiplyValue id (open attribute), 393	ff.evaluator.protocols.yank.SolvationYankProtocol attribute), 555
id (openff.evaluator.protocols.miscellaneous.SubtractValuesid (open	ff.evaluator.server.Batch attribute), 91
	ff.evaluator.workflow.Protocol attribute), 262
id (openff.evaluator.protocols.miscellaneous.WeightByMoleFda&tiper attribute), 403	270
id (openff.evaluator.protocols.openmm.OpenMMEnergyMin üch(squen attribute), 418	ff.evaluator.workflow.schemas.ProtocolGroupSchema attribute), 275
id (openff.evaluator.protocols.openmm.OpenMMEvaluateEndrgi@po attribute), 430	enff.evaluator.workflow.schemas.ProtocolSchema attribute), 273
<pre>id(openff.evaluator.protocols.openmm.OpenMMSimulationidentis</pre>	Eier (openff.evaluator.substances.Amount property), 131
id (openff.evaluator.protocols.paprika.analysis.AnalyzeAPRPdentis attribute), 473	
id (openff.evaluator.protocols.paprika.analysis.ComputeRef actent W attribute), 483	
id (openff.evaluator.protocols.paprika.analysis.ComputeSynivdenyE	
attribute), 478	property), 135
id (openff.evaluator.protocols.paprika.coordinates.AddDumindertoint attribute), 446	property), 126
id (openff.evaluator.protocols.paprika.coordinates.Prepare Autp6val	
attribute), 436 id (openff.evaluator.protocols.paprika.coordinates.PrepareReleaseC	openff.evaluator.datasets.curation.components.freesolv),
	FreeSolvSchema (class in
id (openff.evaluator.protocols.paprika.restraints.ApplyRestraints attribute), 467	openff.evaluator.datasets.curation.components.freesolv), 172
id (openff.evaluator.protocols.paprika.restraints.GenerateAltmploRes	FheinwoMLData (class in
attribute), 451	openff.evaluator.datasets.curation.components.thermoml),
id (openff.evaluator.protocols.paprika.restraints.GeneratePullRestra attribute), 457 Import	unus4 ThermoMLDataSchema (<i>class in</i>
id (openff.evaluator.protocols.paprika.restraints.GenerateReleaseRe	
attribute), 462	173
id (openff.evaluator.protocols.reweighting.BaseEvaluateEneingdex() attribute), 498) (openff.evaluator.storage.attributes.FilePath method), 250
id (openff.evaluator.protocols.reweighting.BaseMBARProto En equal	
attribute), 504	openff.evaluator.workflow.attributes), 281
id (openff.evaluator.protocols.reweighting.ConcatenateObsemphtes	
attribute), 493	(openff.evaluator.protocols.analysis.DecorrelateTrajectory

attribute), 319	<i>method</i>), 250
input_coordinate_file	<pre>isdigit() (openff.evaluator.storage.attributes.FilePath</pre>
(openff.evaluator.protocols.openmm.OpenMMEncolor)	
attribute), 418	<pre>isidentifier() (openff.evaluator.storage.attributes.FilePath</pre>
input_coordinate_file	<i>method</i>), 250
(openff.evaluator.protocols.openmm.OpenMMSir	n usdiomer() (openff.evaluator.storage.attributes.FilePath
attribute), 424	<i>method</i>), 250
<pre>input_coordinate_file</pre>	<pre>isnumeric() (openff.evaluator.storage.attributes.FilePath</pre>
(openff.evaluator.protocols.simulation.BaseEnerg	gyMinimisa ttiet hod), 250
attribute), 518	<pre>isprintable() (openff.evaluator.storage.attributes.FilePath</pre>
<pre>input_coordinate_file</pre>	<i>method</i>), 251
(openff.evaluator.protocols.simulation.BaseSimu	aisspace() (openff.evaluator.storage.attributes.FilePath
attribute), 525	<i>method</i>), 251
input_coordinate_path	<pre>istitle() (openff.evaluator.storage.attributes.FilePath</pre>
(openff.evaluator.protocols.paprika.coordinates.	
attribute), 444	isupper() (openff.evaluator.storage.attributes.FilePath
input_coordinate_paths	method), 251
	aitz Engedt wiesenff.evaluator.utils.observables.Observable Frame
attribute), 486	method), 570
<pre>input_observables(openff.evaluator.protocols.analysis.</pre>	
attribute), 324	J
	s. ፘኇ፼_Gradisน හ (openff.evaluator.backends.dask.BaseDaskJobQueueBacke
attribute), 369	method), 209
	ingo (2099) ingo oscillation backends.dask.DaskLSF Backend
attribute), 490	method), 212
	memoa), 212 puso <u>EilseSupt(</u> GCBB2Rglevaluator.backends.dask.DaskPBSBackend
attribute), 405	method), 214
	ngtestAddDumpspetfff.evaluator.storage.attributes.FilePath
attribute), 444	method), 251
	тетоа, 251 птоАнуу Repting.evaluator.utils.observables.ObservableArray
attribute), 465	class method), 568
input_trajectory_path	
(openff.evaluator.protocols.analysis.Decorrelate	join() (openff.evaluator.utils.observables.ObservableFrame
attribute), 319	
input_trajectory_paths	json() (<i>openff.evaluator.attributes.AttributeClass</i>
(openff evaluator protocols reweighting Concater	method), 565 https://www.science.client.ConnectionOptions
attribute), 486	
input_value (openff.evaluator.protocols.miscellaneous.D	method), 81
attribute), 410	
InputAttribute (class in	json() (openff.evaluator.client.RequestOptions method),
openff.evaluator.workflow.attributes), 282	
inputs (openff.evaluator.workflow.schemas.ProtocolGroup	json() (openff.evaluator.client.RequestResult method),
attribute), 275	
inputs (openff.evaluator.workflow.schemas.ProtocolSchen	json() (openff.evaluator.datasets.CalculationSource
attribute), 273	
inverse_beta (openff.evaluator.thermodynamics.Thermo	json() (openff.evaluator.datasets.MeasurementSource
property), 137	
isalnum() (openff.evaluator.storage.attributes.FilePath	json() (openff.evaluator.datasets.PhysicalProperty
	method), 95
<pre>method), 250 isalpha() (openff.evaluator.storage.attributes.FilePath</pre>	json() (openff.evaluator.datasets.PhysicalPropertyDataSet
method), 250	<i>method</i>), 141
isascii() (openff.evaluator.storage.attributes.FilePath	json() (openff.evaluator.datasets.Source method), 97
	json() (openff.evaluator.datasets.taproom.TaproomDataSet
method), 250	method), 149
isdecimal() (openff.evaluator.storage.attributes.FilePath	

json()	(openff.evaluator.datasets.taproom.TaproomSource jsor method), 151	() (openff.evaluator.protocols.coordinates.BuildDockedCoordinates method), 345
json()	(openff.evaluator.datasets.thermoml.ThermoMLDat gSer method), 144	() (openff.evaluator.protocols.coordinates.SolvateExistingStructure method), 339
json()	(openff.evaluator.forcefield.ForceFieldSource jsor method), 181	() (openff.evaluator.protocols.forcefield.BaseBuildSystem method), 350
json()	(openff.evaluator.forcefield.LigParGenForceFieldSojuscar method), 186	() (openff.evaluator.protocols.forcefield.BuildLigParGenSystem method), 361
json()	(openff.evaluator.forcefield.SmirnoffForceFieldSourgsor method), 183	() (openff.evaluator.protocols.forcefield.BuildSmirnoffSystem method), 355
json()	(openff.evaluator.forcefield.TLeapForceFieldSourcejsor method), 184	() (openff.evaluator.protocols.forcefield.BuildTLeapSystem method), 367
json()	(openff.evaluator.layers.CalculationLayerResult jsor method), 190	() (openff.evaluator.protocols.gradients.ZeroGradients method), 372
	<i>method</i>), 192	() (openff.evaluator.protocols.groups.ConditionalGroup method), 378
	<i>method</i>), 202	a () (openff.evaluator.protocols.groups.ConditionalGroup.Condition method), 375
	<i>method</i>), 198	n() (openff.evaluator.protocols.miscellaneous.AddValues method), 383
json()	(openff.evaluator.layers.workflow.WorkflowCalculat jyst) method), 196	xh@mupenff.evaluator.protocols.miscellaneous.DivideValue method), 398
json()	(openff.evaluator.properties.Density method), j sor 102	() (openff.evaluator.protocols.miscellaneous.DummyProtocol method), 413
json()	(openff.evaluator.properties.DielectricConstant jsor method), 109	() (openff.evaluator.protocols.miscellaneous.FilterSubstanceByRole method), 408
json()	(openff.evaluator.properties.EnthalpyOfMixing jsor method), 113	() (openff.evaluator.protocols.miscellaneous.MultiplyValue method), 393
json()	(openff.evaluator.properties.EnthalpyOfVaporizationjsor method), 116	() (openff.evaluator.protocols.miscellaneous.SubtractValues method), 388
json()	(<i>openff.evaluator.properties.ExcessMolarVolume</i> jsor <i>method</i>), 106	() (openff.evaluator.protocols.miscellaneous.WeightByMoleFraction method), 403
json()	(openff.evaluator.properties.HostGuestBindingAffinijsor method), 123	() (openff.evaluator.protocols.openmm.OpenMMEnergyMinimisation method), 418
json()	(openff.evaluator.properties.SolvationFreeEnergy jsor method), 119	() (openff.evaluator.protocols.openmm.OpenMMEvaluateEnergies method), 430
json()	(openff.evaluator.protocols.analysis.AverageDielectrijs6x method), 301	אָלָאָל(מון enff.evaluator.protocols.openmm.OpenMMSimulation method), 424
json()	(openff.evaluator.protocols.analysis.AverageFreeEngissie method), 307	(<i>openff.evaluator.protocols.paprika.analysis.AnalyzeAPRPhase method</i>), 473
json()	(openff.evaluator.protocols.analysis.AverageObserv glsker method), 295	 (openff.evaluator.protocols.paprika.analysis.ComputeReferenceWor method), 483
json()	(openff.evaluator.protocols.analysis.BaseAverageObjsor method), 290	ADL openff.evaluator.protocols.paprika.analysis.ComputeSymmetryCon method), 478
json()	(openff.evaluator.protocols.analysis.BaseDecorrelatigsor method), 317	Moc (bpenff.evaluator.protocols.paprika.coordinates.AddDummyAtoms method), 446
json()	(openff.evaluator.protocols.analysis.ComputeDipole]son method), 312	n(a)(sopenff.evaluator.protocols.paprika.coordinates.PreparePullCoord method), 436
json()	(openff.evaluator.protocols.analysis.DecorrelateObsjeson method), 327	Klesopenff.evaluator.protocols.paprika.coordinates.PrepareReleaseCo method), 441
json()		r(y) (openff.evaluator.protocols.paprika.restraints.ApplyRestraints method), 467
json()		sPa(dpno)[f.evaluator.protocols.paprika.restraints.GenerateAttachRestra method), 451

json()	(openff.evaluator.protocols.paprika.restraints.Gene method), 457		Restetlints), 237 (openff.evaluator.substances.Amount method),
json()	$(open {\it ff.evaluator.protocols.paprika.restraints.Generation of the second s$	erateRelea	useR2straints
:()	method), 462	json()	(openff.evaluator.substances.Component
json()	(openff.evaluator.protocols.reweighting.BaseEvalue		
icon()	method), 498	json()	(openff.evaluator.substances.ExactAmount
JSON()	(openff.evaluator.protocols.reweighting.BaseMBAF	json()	
icon()	method), 504 (openff.evaluator.protocols.reweighting.Concatena	-	(openff.evaluator.substances.MoleFraction
JS011()	(openjj.evalualor.protocols.reweighting.Concalena method), 493		(openff.evaluator.substances.Substance method),
ison()	(openff.evaluator.protocols.reweighting.Concatena		
J3011()	(openy).evaluator.protocois.reweighting.concurent method), 488		openff.evaluator.thermodynamics.ThermodynamicState
ison()	(openff.evaluator.protocols.reweighting.ReweightD		
J3011()	method), 515		(openff.evaluator.utils.exceptions.EvaluatorException
ison()	(openff.evaluator.protocols.reweighting.ReweightO		
J3011()	method), 509		(openff.evaluator.workflow.Protocol method),
ison()	(openff.evaluator.protocols.simulation.BaseEnergy		
J3011()	method), 521	json()	(openff.evaluator.workflow.ProtocolGroup
ison()	(openff.evaluator.protocols.simulation.BaseSimula		method), 270
J3011()	method), 528		openff.evaluator.workflow.schemas.ProtocolGroupSchema
ison()	(openff.evaluator.protocols.storage.UnpackStoredS		
J3011()	method), 534		openff.evaluator.workflow.schemas.ProtocolReplicator
ison()	(openff.evaluator.protocols.yank.BaseYankProtocol	-	method), 278
J0011()	method), 540		(openff.evaluator.workflow.schemas.ProtocolSchema
ison()	(openff.evaluator.protocols.yank.LigandReceptorYc	-	
J3011()	method), 547		openff.evaluator.workflow.schemas.WorkflowSchema
ison()	(openff.evaluator.protocols.yank.SolvationYankPro		method), 280
J0011()	method), 556	json()	(openff.evaluator.workflow.WorkflowException
ison()	(openff.evaluator.server.Batch method), 92	J0011()	method), 257
	(openff.evaluator.storage.data.BaseSimulationData	ison()	(openff.evaluator.workflow.WorkflowResult
J ()	<i>method</i>), 227	J. ()	<i>method</i>), 260
json()	(openff.evaluator.storage.data.BaseStoredData		
J ()	<i>method</i>), 220	K	
json()	(openff.evaluator.storage.data.ForceFieldData	kevs()(openff.evaluator.utils.observables.ObservableFrame
5	method), 223	KCy5()(method), 570
json()	(openff.evaluator.storage.data.HashableStoredData	а	memory, 570
5	<i>method</i>), 221	L	
json()	(openff.evaluator.storage.data.ReplaceableData		acto col (on out and uston work for utile Proto col Dath
-	method), 225	_	rotocol (openff.evaluator.workflow.utils.ProtocolPath
json()	(openff.evaluator.storage.data.StoredFreeEnergyD	ata	property), 285 purce (openff.evaluator.forcefield.TLeapForceFieldSource
	1101104, 255		(100)
json()	(openff.evaluator.storage.data.StoredSimulationDa	talet he	property), 184 and_value(openff.evaluator.protocols.groups.ConditionalGroup.C
	<i>method</i>), 230	IeIt_IIa	attribute), 375
json()	(openff.evaluator.storage.query.BaseDataQuery	ligand	electrostatic_lambdas
-	method), 236	11yanu_	(openff.evaluator.protocols.yank.LigandReceptorYankProtocol
json()	(openff.evaluator.storage.query.BaseSimulationDa	taQuery	attribute), 544
-	<i>method</i>), 241		residue_name
json()	(openff.evaluator.storage.query.ForceFieldQuery	11yanu_	(openff.evaluator.protocols.coordinates.BuildDockedCoordinates
	<i>method</i>), 239		(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
json()	(openff.evaluator.storage.query.FreeEnergyDataQu	uaryand	residue name
	<i>method</i>), 246	11yanu_	(openff.evaluator.protocols.yank.LigandReceptorYankProtocol
json()	(openff.evaluator.storage.query.SimulationDataQu	ery	attribute), 543
	method), 243		steric_lambdas
json()	(openff.evaluator.storage.query.SubstanceQuery	rryanu_	(openff.evaluator.protocols.yank.LigandReceptorYankProtocol
			(openiji, evaluator, protocolis, yank. Ligunakeepior ranki 1010001

attribute), 544	<pre>merge() (openff.evaluator.protocols.analysis.AverageObservable</pre>
ligand_substance(openff.evaluator.protocols.coordinate	es.BuildDo ckethodo rdAtates
attribute), 342	<pre>merge() (openff.evaluator.protocols.analysis.BaseAverageObservable</pre>
LigandReceptorYankProtocol (class in	<i>method</i>), 290
openff.evaluator.protocols.yank), 541	merge() (openff.evaluator.protocols.analysis.BaseDecorrelateProtocol
LigandReceptorYankProtocol.RestraintType	<i>method</i>), 317
(class in openff.evaluator.protocols.yank), 543	<pre>merge() (openff.evaluator.protocols.analysis.ComputeDipoleMoments</pre>
LigParGenForceFieldSource (class in	<i>method</i>), 312
	<pre>merge() (openff.evaluator.protocols.analysis.DecorrelateObservables</pre>
LigParGenForceFieldSource.ChargeModel (class in	<i>method</i>), 327
	merge() (open ff. evaluator. protocols. analysis. Decorrelate Trajectory
ljust() (openff.evaluator.storage.attributes.FilePath	<i>method</i>), 322
<i>method</i>), 251	merge() (openff.evaluator.protocols.coordinates.BuildCoordinatesPackmol
<pre>load_restraints() (openff.evaluator.protocols.paprika.r</pre>	restraints.AppellyRd9trabbts
	$\verb merge() (open ff. evaluator. protocols. coordinates. BuildDockedCoordinates) $
LocalFileStorage (class in openff.evaluator.storage),	<i>method</i>), 345
	<pre>merge() (openff.evaluator.protocols.coordinates.SolvateExistingStructure</pre>
lower() (openff.evaluator.storage.attributes.FilePath	<i>method</i>), 339
	<pre>merge() (openff.evaluator.protocols.forcefield.BaseBuildSystem</pre>
<pre>lstrip() (openff.evaluator.storage.attributes.FilePath</pre>	<i>method</i>), 351
<i>method</i>), 251	<pre>merge() (openff.evaluator.protocols.forcefield.BuildLigParGenSystem</pre>
N /	<i>method</i>), 361
M	<pre>merge() (openff.evaluator.protocols.forcefield.BuildSmirnoffSystem</pre>
<pre>maketrans() (openff.evaluator.storage.attributes.FilePath</pre>	
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	<pre>merge() (openff.evaluator.protocols.forcefield.BuildTLeapSystem</pre>
mass_density (openff.evaluator.protocols.coordinates.Bui	ildCoordintilesPeeckmbt7
attribute), 330	merge() (openff.evaluator.protocols.gradients.ZeroGradients
mass_density (openff.evaluator.protocols.coordinates.Sol	lvateExistingsthaefure72
	merge() (openff.evaluator.protocols.groups.ConditionalGroup
max_iterations (openff.evaluator.protocols.groups.Cond	ditionalGromethod), 376
	merge() (openff.evaluator.protocols.miscellaneous.AddValues
<pre>max_iterations(openff.evaluator.protocols.openmm.Ope</pre>	enMMEner#44Hindinisation
	merge() (openff.evaluator.protocols.miscellaneous.DivideValue
<pre>max_iterations(openff.evaluator.protocols.simulation.Be</pre>	BaseEnergyMathmasation
	merge() (openff.evaluator.protocols.miscellaneous.DummyProtocol
<pre>max_molecules(openff.evaluator.protocols.coordinates.Bu</pre>	BuildCoordinatesPelokmol
	merge() (openff.evaluator.protocols.miscellaneous.FilterSubstanceByRole
<pre>max_molecules(openff.evaluator.protocols.coordinates.Sc</pre>	olvateExistingSipaduare ⁸
	merge() (openff.evaluator.protocols.miscellaneous.MultiplyValue
maximum_data_points	<i>method</i> ), 393
(openff.evaluator.layers.reweighting.Reweighting	STATES () (openff.evaluator.protocols.miscellaneous.SubtractValues
attribute), 201	method), 388
incubul emericoboul ee (clubs) in	merge() (openff.evaluator.protocols.miscellaneous.WeightByMoleFraction
openff.evaluator.datasets), 99	<i>method</i> ), 403
	<pre>smerge() (openff.evaluator.protocols.openmm.OpenMMEnergyMinimisation</pre>
method), 139	<i>method</i> ), 418
<pre>merge() (openff.evaluator.datasets.taproom.TaproomDatas</pre>	<pre>smerge() (openff.evaluator.protocols.openmm.OpenMMEvaluateEnergies</pre>
<i>method</i> ), 149	method), 430
<pre>merge() (openff.evaluator.datasets.thermoml.ThermoMLD</pre>	masse () (openff.evaluator.protocols.openmm.OpenMMSimulation
method) 144	method), 424
<pre>merge() (openff.evaluator.protocols.analysis.AverageDiele</pre>	econfectionenff.evaluator.protocols.paprika.analysis.AnalyzeAPRPhase
method), 301	method), $4/3$
	Emerges() (openff.evaluator.protocols.paprika.analysis.ComputeReferenceWe
<i>method</i> ), 307	<i>method</i> ), 483

merge()	(openff.evaluator.protocols.paprika.analysis.Comp method), 478	ppmetSydarbat(vyConffactationator.properties.EnthalpyOfVaporization attribute), 117
merge()		AdullandanaA(topenff.evaluator.properties.ExcessMolarVolume attribute), 106
merge()		PrapuedPatlaCopedificateduator.properties.HostGuestBindingAffinity attribute), 124
merge()		PraptredReded(sqConffreinlutestor.properties.SolvationFreeEnergy attribute), 120
merge()		p Mode Frances in openff.evaluator.substances), 134
merge()	(openff.evaluator.protocols.paprika.restraints.Gen method), 452	n <b>wostAindoResataions()</b> (openff.evaluator.storage.data.BaseSimulationData class method), 227
merge()	(openff.evaluator.protocols.paprika.restraints.Ger method), 457	n <b>wostPillRestmainion()</b> (openff.evaluator.storage.data.ReplaceableData class method), 224
merge()	(openff.evaluator.protocols.paprika.restraints.Gen method), 462	n <b>wostReihefsethesticon(s)</b> (openff.evaluator.storage.data.StoredFreeEnergyDc class method), 232
-	method), 498	<pre>lumosFineingficormation() (openff.evaluator.storage.data.StoredSimulationDat class method), 229</pre>
merge()	(openff.evaluator.protocols.reweighting.BaseMBA method), 504	A Relational er (openff.evaluator.protocols.miscellaneous.MultiplyValue attribute), 391
merge()	(openff.evaluator.protocols.reweighting.Concatent method), 493	ad Mar Dhig Jy Blackue (class in openff.evaluator.protocols.miscellaneous),
-	(openff.evaluator.protocols.reweighting.Concatent method), 488	
	(openff.evaluator.protocols.reweighting.Reweight) method), 515	n_microstates (openff.evaluator.protocols.paprika.analysis.ComputeSym
merge()	(openff.evaluator.protocols.reweighting.Reweight) method), 509	Observableattribute), 475 n_pull_windows (openff.evaluator.protocols.paprika.coordinates.Prepare.
merge()	(openff.evaluator.protocols.simulation.BaseEnerg method), 521	
-	(openff.evaluator.protocols.simulation.BaseSimula method), 528	number_of_components
merge()	(openff.evaluator.protocols.storage.UnpackStored, method), 534	lSimulation <b>Daten</b> ff.evaluator.substances.Substance prop- erty), 126
	(openff.evaluator.protocols.yank.BaseYankProtocomethod), 540	(openff.evaluator.protocols.yank.BaseYankProtocol
	(openff.evaluator.protocols.yank.LigandReceptorY method), 547	YankProtocaltribute), 537 number_of_equilibration_iterations
merge()	(openff.evaluator.protocols.yank.SolvationYankPromethod), 556	
merge()	(openff.evaluator.workflow.Protocol method), 263	<pre>number_of_equilibration_iterations</pre>
merge()	(openff.evaluator.workflow.ProtocolGroup method), 269	attribute), 556 number_of_gpus (openff.evaluator.backends.ComputeResources
MergeBe	haviour (class in	property), 205
	openff.evaluator.workflow.attributes), 281	$\verb+number_of_gpus(\textit{openff.evaluator.backends.QueueWorkerResources})$
metadat	attribute), 94	<pre>property), 206 number_of_iterations</pre>
	a (openff.evaluator.properties.Density attribute), 103	(openff.evaluator.protocols.yank.BaseYankProtocol attribute), 537
metadat	a (openff.evaluator.properties.DielectricConstant attribute), 110	<pre>number_of_iterations     (openff.evaluator.protocols.yank.LigandReceptorYankProtocol</pre>
metadat	a (openff.evaluator.properties.EnthalpyOfMixing attribute), 113	<i>attribute</i> ), 548 number_of_iterations

(openff.evaluator.protocols.yank.SolvationYankF	Protocol	attribute), 518
attribute), 556	output_	coordinate_file
<pre>number_of_ligand_conformers</pre>		(openff.evaluator.protocols.simulation.BaseSimulation
(open ff. evaluator. protocols. coordinates. Build Do	ckedCoord	li <b>attu</b> aibute), 525
attribute), 342	output_	.coordinate_path
number_of_molecules		(openff.evaluator.protocols.paprika.coordinates.AddDummyAtom
(openff.evaluator.storage.data.StoredSimulation)	Data	attribute), 444
attribute), 229		coordinate_path
number_of_molecules	• -	(openff.evaluator.protocols.paprika.coordinates.PreparePullCoor
(openff.evaluator.storage.query.SimulationDatag	Juerv	attribute), 436
attribute), 243		coordinate_path
		s (openff.evaluator.protocols.paprika.coordinates.PrepareReleaseC
property), 205		attribute), 441
number_of_threads (openff.evaluator.backends.QueueW	ochert Rest o	
property), 207	o a to e presa	(openff.evaluator.protocols.reweighting.ConcatenateTrajectories
<i>property)</i> , 207		attribute), 486
0	output	frequency (openff.evaluator.protocols.openmm.OpenMMSimulat
-	output_	attribute), 425
Observable (class in openff.evaluator.utils.observables),	output	frequency (openff.evaluator.protocols.simulation.BaseSimulation
566		
observable (openff.evaluator.protocols.analysis.Averaged	Observable	number_of_molecules
attribute), 293		
	eightObser	v(appenff.evaluator.protocols.coordinates.BuildCoordinatesPackmo
attribute), 506	~··+~··+	attribute), 331
ObservableArray (class in	output_	number_of_molecules
openff.evaluator.utils.observables), 567		(openff.evaluator.protocols.coordinates.SolvateExistingStructure
ObservableFrame (class in		attribute), 339
openff.evaluator.utils.observables), 569		observables (openff.evaluator.protocols.analysis.DecorrelateOb
observables (openff.evaluator.protocols.openmm.OpenM	IMSimulat	i officience), 325
attribute), 425	-	observables (openff.evaluator.protocols.gradients.ZeroGradient.
$observables ({\it openff.evaluator.protocols.simulation.Base}$	Simulation	attribute), 370
attribute), 526		observables (openff.evaluator.protocols.openmm.OpenMMEvalu
observables (openff.evaluator.protocols.storage.Unpack	StoredSimu	llattichterata 430
attribute), 531		observables (openff.evaluator.protocols.reweighting.BaseEvaluation)
observables (openff.evaluator.storage.data.StoredSimula	itionData	attribute), 496
attribute), 229	output_	observables (openff.evaluator.protocols.reweighting.Concatenat
ObservableType (class in		attribute), 491
openff.evaluator.utils.observables), 568	-	substance (openff.evaluator.protocols.coordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCoordinates.BuildCo
offset (openff.evaluator.protocols.paprika.coordinates.A	ddDummyA	Aftribute), 331
attribute), 444	output_	substance (openff.evaluator.protocols.coordinates.SolvateExistin
OpenMMEnergyMinimisation (class in		attribute), 339
openff.evaluator.protocols.openmm), 415	output_	$\verb system(openff.evaluator.protocols.paprika.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.AddDummarks.coordinates.coordinates.AddDummarks.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordinates.coordina$
OpenMMEvaluateEnergies (class in		attribute), 444
openff.evaluator.protocols.openmm), 427	output_	$\verb system(openff.evaluator.protocols.paprika.restraints.ApplyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.applyRestraints.apply$
OpenMMSimulation (class in		attribute), 465
openff.evaluator.protocols.openmm), 420	output_	trajectory_path
options (openff.evaluator.server.Batch attribute), 91		$(open {\it ff.evaluator.protocols.analysis.Decorrelate Trajectory} )$
<pre>output_coordinate_file</pre>		attribute), 320
(openff.evaluator.protocols.openmm.OpenMME	1ePytAPutin	h; <b>a</b> jectory_path
attribute), 418	0.2	(open ff. evaluator. protocols. reweighting. Concatenate Trajectories
output coordinate file		attribute), 486
- (openff.evaluator.protocols.openmm.OpenMMSi	malttaht_	value (openff.evaluator.protocols.miscellaneous.DummyProtocol
attribute), 425		attribute), 410
output coordinate file		ttribute (class in
(openff.evaluator.protocols.simulation.BaseEner	gyMinimis	dapgnff.evaluator.workflow.attributes), 282

-		( <i>opt</i> enff.evaluator.protocols.paprika.analysis.AnalyzeAPRPhase property), 473
		(openff.evaluator.protocols.paprika.analysis.ComputeReferenceWo
-		property), 483
		(openff.evaluator.protocols.paprika.analysis.ComputeSymmetryCo
-		property), 478
		<i>(openff.evaluator.protocols.paprika.coordinates.AddDummyAtoms</i>
-		property), 447
		topenff.evaluator.protocols.paprika.coordinates.PreparePullCoord
		property), 436
		(openff.evaluator.protocols.paprika.coordinates.PrepareReleaseC
		property), 441
		copenff.evaluator.protocols.paprika.restraints.ApplyRestraints
		property), 468
		(openff.evaluator.protocols.paprika.restraints.GenerateAttachRest
		property), 452
outputs	(openff.evaluator.protocols.coordinates.BuildCoordintpestBa	(dpen/ff.evaluator.protocols.paprika.restraints.GeneratePullRestra
	property), 333	property), 457
outputs	(openff.evaluator.protocols.coordinates.BuildDock <b>edCpat</b> di	$hap\!en$ ff. evaluator. protocols. paprika. restraints. Generate Release Rele
		property), 462
outputs	(openff.evaluator.protocols.coordinates.SolvateExioningStrs	(toppenff.evaluator.protocols.reweighting.BaseEvaluateEnergies
	property), 339	property), 499
outputs	(openff.evaluator.protocols.forcefield.BaseBuildSystamputs	(openff.evaluator.protocols.reweighting.BaseMBARProtocol
	property), 351	property), 504
outputs	(openff.evaluator.protocols.forcefield.BuildLigParGurtfyutes)	(openff.evaluator.protocols.reweighting.ConcatenateObservables
		property), 493
outputs	(openff.evaluator.protocols.forcefield.BuildSmirnoffSitsperts	(openff.evaluator.protocols.reweighting.ConcatenateTrajectories
		property), 488
		$(open {\it ff.evaluator.protocols.reweighting.ReweightDielectricConstant of the constant of th$
		property), 515
		(openff.evaluator.protocols.reweighting.ReweightObservable
		property), 509
		(openff.evaluator.protocols.simulation.BaseEnergyMinimisation
		property), 521
	(openff.evaluator.protocols.miscellaneous.AddValuerstputs	
		property), 528
		(openff.evaluator.protocols.storage.UnpackStoredSimulationData
		property), 534
-	(openff.evaluator.protocols.miscellaneous.DummyButtpatk	
		property), 540
		(Rplanff.evaluator.protocols.yank.LigandReceptorYankProtocol
		property), 548
	(openff.evaluator.protocols.miscellaneous.Multiply divigents	
		property), 556
	(openff.evaluator.protocols.miscellaneous.SubtractStarlagests	
	property), 389	262
	(openff.evaluator.protocols.miscellaneous.WeightByWfplaFsa	
		property), 268
		istaticatore (openff.evaluator.workflow.schemas.WorkflowSchema
		attribute), 279
	(openff.evaluator.protocols.openmm.OpenMMEvaloratafirtas; property) 430	
	property), 430 (openff.evaluator.protocols.openmm.OpenMMSimulation	property), 255
	property), 425	
	property, 120	

## р

P				<i>method</i> ), 100
<pre>parameter_gradient_keys</pre>		parse_		(openff.evaluator.datasets.PhysicalProperty
(openff.evaluator.server.Batch	attribute),			method), 95
91		parse_		(openff.evaluator.datasets.PhysicalPropertyDataSet
ParameterGradient (class	in			<i>method</i> ), 141
openff.evaluator.forcefield), 188		parse_		(openff.evaluator.datasets.Source class
ParameterGradientKey (class	in			pd), 97
openff.evaluator.forcefield), 187		parse_		(openff.evaluator.datasets.taproom.TaproomDataSet
parameterized_system		n2nc0	ciass	method), 149
(openff.evaluator.protocols.analysis.	ComputeDip	oleMom	ants class	(openff.evaluator.datasets.taproom.TaproomSource method), 151
attribute), 309		narco		(openff.evaluator.datasets.thermoml.ThermoMLDataSet
parameterized_system				method), 144
(openff.evaluator.protocols.forcefield	d.BaseBuildS	ystem narse		(openff.evaluator.forcefield.ForceFieldSource
attribute), 348		pui se_		method), 181
parameterized_system		parse.	ison()	(openff.evaluator.forcefield.LigParGenForceFieldSource
(openff.evaluator.protocols.forcefield	a.BuilaLigPa	rGensyst	class	method), 187
attribute), 361		parse		(openff.evaluator.forcefield.SmirnoffForceFieldSource
<pre>parameterized_system     (openff.evaluator.protocols.forcefield)</pre>	d DuildSminn			
attribute), 356	и.Биназтит	parse_	; json()	(openff.evaluator.forcefield.TLeapForceFieldSource
narameterized system			class	<i>method</i> ), 184
(openff evaluator protocols forcefield	d RuildTLear	marse_	json()	(openff.evaluator.layers.CalculationLayerResult
attribute), 367	a.Dana i Leap	<i>bysicm</i>	class	method), 191
parameterized_system		parse_	json()	(openff.evaluator.layers.CalculationLayerSchema
(openff.evaluator.protocols.openmm	.OpenMMEn	ergvMin	imEstario	pethod), 192
attribute), 419	- I	parse_	Json()	(openjj.evalualor.layers.reweignling.KeweignlingSchema
namamatanizad auctom			class	<i>method</i> ), 202
(openff.evaluator.protocols.openmm	.OpenMMEv	alittieEn	jsqp()	(openff.evaluator.layers.simulation.SimulationSchema
attribute), 430	*		ciuss	<i>meinoa</i> ), 198
parameterized_system		parse_		(openff.evaluator.layers.workflow.WorkflowCalculationSch
(openff.evaluator.protocols.openmm	.OpenMMSir	nulation		method), 196
attribute), 425		parse_		
parameterized_system			class	method), 103
(openff.evaluator.protocols.reweight	ing.BaseEva	lıtateÉfie	lson ()	(openff.evaluator.properties.DielectricConstant mathed), 110
attribute), 496			ciuss	method), 110 (openff.evaluator.properties.EnthalpyOfMixing
parameterized_system				
(openfi.evaluator.protocols.simulatio	on.BaseEnerg	gyMinimi narse	isatton ^o	(openff.evaluator.properties.EnthalpyOfVaporization
attribute), 518		parse_	-	method), 117
parameterized_system		narse	ison()	(openff.evaluator.properties.ExcessMolarVolume
	on.BaseSimu	<i>lation</i> ⁻ -	class	method), 106
<pre>attribute), 525 parse_json() (openff.evaluator.attributes.At</pre>	twibuteClass	parse		(openff.evaluator.properties.HostGuestBindingAffinity
class method) 565			class	<i>method</i> ), 124
narse ison() (openff evaluator client Conne	octionOntion	parse_	json()	(openff.evaluator.properties.SolvationFreeEnergy
class method), 81	cuonopuon	5 -	class	method), 120
parse_json() (openff.evaluator.client.Req	uest class	parse_	json()	$(open {\it ff.evaluator.protocols.analysis.Average Dielectric Construction of the cons$
method), 83			class	<i>method</i> ), 301
<pre>parse_json() (openff.evaluator.client.Requ</pre>	uestOptions	parse_	json()	$(open {\it ff.evaluator.} protocols. analysis. A verage Free Energies$
class method), 85			class	method), 307
<pre>parse_json() (openff.evaluator.client.Re</pre>	questResult	parse_		$(open {\it ff.evaluator.protocols.analysis.} Average Observable$
class method) 87	-		class	<i>method</i> ), 296
<pre>parse_json() (openff.evaluator.datasets.Cal</pre>	culationSour	$_{c}$ parse_	json()	(openff.evaluator.protocols.analysis.BaseAverageObservab
class method) 08			ciass	methoa), 290
<pre>parse_json() (openff.evaluator.datasets.Med</pre>	asurementSo	ufcerse_	]son()	$(open {\it ff.evaluator.protocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analysis.BaseDecorrelateProtocols.analys$

class method), 317	class method), 447
parse_json() (openff.evaluator.protocols.analysis.Compu <b>matipe</b> le <b>j</b>	<b>Som</b> (h)(copenff.evaluator.protocols.paprika.coordinates.PreparePu
class method), 312	class method), 436
parse_json() (openff.evaluator.protocols.analysis.Decorr <b>clars@</b> by	${\tt sonkle} (open ff. evaluator. protocols. paprika. coordinates. Prepare Relation of the state of the state$
class method), 327	class method), 441
parse_json() (openff.evaluator.protocols.analysis.Decorr <b>clareSte</b> ajj	<b>som(y)</b> (openff.evaluator.protocols.paprika.restraints.ApplyRestrai
class method), 322	class method), 468
parse_json() (openff.evaluator.protocols.coordinates.Buildfrædj	BorsPachpeolff.evaluator.protocols.paprika.restraints.GenerateAtta
class method), 334	class method), 452
parse_json() (openff.evaluator.protocols.coordinates.Buildarsted	
class method), 345	class method), 457
parse_json() (openff.evaluator.protocols.coordinates.Solvpædfseistj.	
class method), 340	class method), 462
parse_json() (openff.evaluator.protocols.forcefield.BaseBpildSestej	
class method), 351	class method), 499
parse_json() (openff.evaluator.protocols.forcefield.BuildIpgPseGg	
class method), 361	class method), 504
parse_json() (openff.evaluator.protocols.forcefield.BuildSpairseffS	
class method), 356	class method), 493
parse_json() (openff.evaluator.protocols.forcefield.BuildTflamp6yg	
class method), 367	class method), 488
parse_json() (openff.evaluator.protocols.gradients.ZeroGpadisetsj	
class method), 372	class method), 515
parse_json() (openff.evaluator.protocols.groups.Conditio <b>pak(Se</b> uj	
class method), 379	class method), 509
parse_json() (openff.evaluator.protocols.groups.Conditio <b>palGre</b> ug	
class method), 375	class method), 521
parse_json() (openff.evaluator.protocols.miscellaneous.A <b>ddVsk</b> æj	
class method), 384	class method), 528
parse_json() (openff.evaluator.protocols.miscellaneous.Dpadb&alj	<b>gon()</b> (openff.evaluator.protocols.storage.UnpackStoredSimulation
class method), 398	class method), 534
parse_json()(openff.evaluator.protocols.miscellaneous.DpansePj	rest (appenff.evaluator.protocols.yank.BaseYankProtocol
class method), 413	class method), 540
parse_json() (openff.evaluator.protocols.miscellaneous.F <b>pærSa</b> bj	######################################
class method), 408	class method), 548
parse_json() (openff.evaluator.protocols.miscellaneous.Mpd#pby_Vg	stom() (openff.evaluator.protocols.yank.SolvationYankProtocol
class method), 393	class method), 556
parse_json() (openff.evaluator.protocols.miscellaneous.SphatrsetVg	stors() (openff.evaluator.server.Batch class
class method), 389	method), 92
parse_json() (openff.evaluator.protocols.miscellaneous.Weight@yj	som Fraction ff. evaluator. storage. data. Base Simulation Data
<i>class method</i> ), 403	class method), 227
parse_json() (openff.evaluator.protocols.openmm.OpenMp4Eserg)	
class method), 419	class method), 220
parse_json() (openff.evaluator.protocols.openmm.OpenMpaEseluj	
class method), 431	
narse ison() (on anthe avaluator protocols on any On an MAA Some la	class method), 223
	class method), 223 (sion() (openff.evaluator.storage.data.HashableStoredData
class method), 425	class method), 223 (sion() (openff.evaluator.storage.data.HashableStoredData class method), 221
class method), 425 parse_json() (openff.evaluator.protocols.paprika.analysis <b>påmsk</b> yzg	class method), 223 (sion() (openff.evaluator.storage.data.HashableStoredData class method), 221 \$BRP(h(spenff.evaluator.storage.data.ReplaceableData
class method), 425 parse_json() (openff.evaluator.protocols.paprika.analysispanskyzg. class method), 473	class method), 223 sion() (openff.evaluator.storage.data.HashableStoredData class method), 221 sBRP(h(spenff.evaluator.storage.data.ReplaceableData class method), 225
class method), 425 parse_json() (openff.evaluator.protocols.paprika.analysispanskyzۇ. class method), 473 parse_json() (openff.evaluator.protocols.paprika.analysispanskyuj	class method), 223 sion() (openff.evaluator.storage.data.HashableStoredData class method), 221 <b>\$BR(P)</b> (aspenff.evaluator.storage.data.ReplaceableData class method), 225 <b>sRaf(E)</b> e(apeMiffikvaluator.storage.data.StoredFreeEnergyData
<pre>class method), 425 parse_json() (openff.evaluator.protocols.paprika.analysispanskyzė,</pre>	class method), 223 sion() (openff.evaluator.storage.data.HashableStoredData class method), 221 SBR(Ph(spenff.evaluator.storage.data.ReplaceableData class method), 225 sRaf(c)e(speak)(frkvaluator.storage.data.StoredFreeEnergyData class method), 233
class method), 425 parse_json() (openff.evaluator.protocols.paprika.analysispanskyzğ. class method), 473 parse_json() (openff.evaluator.protocols.paprika.analysispanskyuğ class method), 483 parse_json() (openff.evaluator.protocols.paprika.analysispanskyuğ	class method), 223 (sion() (openff.evaluator.storage.data.HashableStoredData class method), 221 <b>\$BR(P)</b> (uspenff.evaluator.storage.data.ReplaceableData class method), 225 <b>\$Raf()</b> (upeNff:kvaluator.storage.data.StoredFreeEnergyData class method), 233 <b>\$Spw()</b> )(typ)(iff:eectlinentor.storage.data.StoredSimulationData
<pre>class method), 425 parse_json() (openff.evaluator.protocols.paprika.analysispanskyzė,</pre>	class method), 223 (sion() (openff.evaluator.storage.data.HashableStoredData class method), 221 <b>\$BRP</b> (h(xpenff.evaluator.storage.data.ReplaceableData class method), 225 <b>\$Raf()</b> (ye(xpeMiff:kvaluator.storage.data.StoredFreeEnergyData class method), 233 <b>\$Sym()</b> (xtpp:Giff:eeculiantor.storage.data.StoredSimulationData class method), 230

<pre>class method), 236 parse_json() (openff.evaluator.storage.query.BaseSimula</pre>		penff.evaluator.properties.	EnthalpyOfVaporiz	ation
class method), 242		ppenff.evaluator.propertie	s ExcessMolarVolu	ne
parse_json() (openff.evaluator.storage.query.ForceFieldy		attribute), 106	S.E.Accessificitar volui	ne
class method), 239		penff.evaluator.properties.	HostGuestRinding	ffinity
parse_json() (openff.evaluator.storage.query.FreeEnergy			nosiGuesiDinuingA	gjinii y
			Soluction Enco Enco	1011
class method), 246		penff.evaluator.properties	s.solvallonr reeller	<i>8y</i>
parse_json() (openff.evaluator.storage.query.Simulation)			· ······:	
class method), 244		penff.evaluator.protocols.p	paprika.anaiysis.And	uyzeAPKPnase
<pre>parse_json() (openff.evaluator.storage.query.SubstanceQ</pre>		attribute), 470		
class method), 237	-	penff.evaluator.protocols.p	oaprika.restraints.Aj	oplyRestraints
parse_json() (openff.evaluator.substances.Amount		attribute), 465		
class method), 132		_property ( <i>openff.evalu</i>	iator.layers.Calcula	tionLayerResult
<pre>parse_json() (openff.evaluator.substances.Component</pre>		attribute), 190		
class method), 130	Physical	Property (class in open	ıff.evaluator.dataset	s),
<pre>parse_json() (openff.evaluator.substances.ExactAmount</pre>		93		
class method), 134	Physical	PropertyDataSet	(class	in
<pre>parse_json() (openff.evaluator.substances.MoleFraction</pre>		openff.evaluator.datasets)		
class method), 136	placehol	der_id(openff.evaluator	r.workflow.schemas.	ProtocolReplicator
<pre>parse_json() (openff.evaluator.substances.Substance</pre>		property), 277		
class method), 128	Placehol	lderValue	(class	in
<pre>parse_json() (openff.evaluator.thermodynamics.Thermodynamics</pre>	dynamicSta	wenff.evaluator.attribute	s), 566	
class method), 138	•	penff.evaluator.utils.obser		rame
<pre>parse_json() (openff.evaluator.utils.exceptions.Evaluator</pre>				
class method), 88	-	in_path() (openff.eva	luator.workflow.utils	s.ProtocolPath
parse_json() (openff.evaluator.workflow.Protocol class		method), 285		
method), 265		() (openff.evaluator.utils.	observables Observa	ableFrame
<pre>parse_json() (openff.evaluator.workflow.ProtocolGroup</pre>		method), 570	Joser vabres. Obser va	ioter tunie
class method), 270		al_energies(openff.eva	luator protocols and	lysis Average Dielect
parse_json() (openff.evaluator.workflow.schemas.Protoco			iuuior.proiocois.una	uysis.AverugeDieleci
	-		luator protocols and	husia Ananaga Ohaam
class method), 275		al_energies( <i>openff.eval</i>	iuator.protocots.and	uysis.AverageObserv
parse_json() (openff.evaluator.workflow.schemas.Protoco	-		1	
class method), 278		al_energies( <i>openff.eva</i>	luator.protocols.and	uysis.BaseAverageOl
<pre>parse_json() (openff.evaluator.workflow.schemas.Protoce</pre>				
class method), 273		ed_charge_model		
<pre>parse_json() (openff.evaluator.workflow.schemas.Workflow</pre>			ld.LigParGenForcel	FieldSource
class method), 280		property), 186		
<pre>parse_json() (openff.evaluator.workflow.WorkflowExcept</pre>	<i>ipn</i> eferre	ed_gpu_toolkit		
class method), 258		(openff.evaluator.backend	ls.ComputeResource	?S
<pre>parse_json() (openff.evaluator.workflow.WorkflowResult</pre>		property), 205		
class method), 260	preferre	ed_gpu_toolkit		
<pre>partition() (openff.evaluator.storage.attributes.FilePath</pre>		(openff.evaluator.backend	ls.QueueWorkerRes	ources
<i>method</i> ), 251		property), 207		
per_thread_memory_limit		PullCoordinates	(class	in
(openff.evaluator.backends.QueueWorkerResourc	-	openff.evaluator.protocol		es),
property), 206		432	1 1	· · ·
phase (openff.evaluator.datasets.PhysicalProperty	Prepare	ReleaseCoordinates	(class	in
attribute), 94	-	openff.evaluator.protocol	<b>`</b>	
phase (openff.evaluator.properties.Density attribute),		438	s.papina.cooranaa	,
103 (openjj.evananor.propernes.Density unribute),		_protocol_id()		
		openff.evaluator.workflow	w utils Protocol Path	
phase (openff.evaluator.properties.DielectricConstant attribute), 110		(openij).evalualor.workjio method), 285	w.utits.1 10100011'011	
			dynamics Thomas d	mamieState
		e (openff.evaluator.thermo	aynamics.1nermoa	ynamicsiale
tribute), 113		attribute), 137		

<pre>properties (openff.evaluator.datasets.PhysicalPropertyD</pre>		<i>attribu</i> 1_sche	,,	enff.e	valuator.workflow.scl	hemas.WorkflowSchema
<pre>properties (openff.evaluator.datasets.taproom.TaproomL</pre>		<i>attribu</i> lGraph		in	openff.evaluator.worl	kflow),
properties (openff.evaluator.datasets.thermoml.ThermoM	ALDataSet	t 266				
<pre>property), 144 properties_by_substance()</pre>	Protoco	267	(class	in	openff.evaluator.worl	KJIOW),
(openff.evaluator.datasets.PhysicalPropertyData	SProtoco	lGroup	Schema		(class	in
<i>method</i> ), 139		openff.	evaluator	r.wor	kflow.schemas), 274	
<pre>properties_by_substance()</pre>	ıSet	284		open	ff.evaluator.workflow	e.utils),
<i>method</i> ), 149	Protoco				(class	in
<pre>properties_by_substance()     (openff.evaluator.datasets.thermoml.ThermoMLL     method), 144</pre>	anno ed co	x	nff.evalud		kflow.schemas), 276 protocols.groups.Con	nditionalGroup
<pre>properties_by_type()</pre>	protoco	ls (d	penff.evd	aluat	or.workflow.Protocol	Graph
(openff.evaluator.datasets.Physical Property Datasets.Physical Property Datasets.Physical Property Datasets	Set	· ·	ty), 266			
method), 140	protoco			aluat	or.workflow.Protocol	Group
properties_by_type() (on outfinite output of the set of	State Co	proper		uator	www.wl.flow Workflow	9409
(openff.evaluator.datasets.taproom.TaproomData method), 149		erty), 2	55			
<pre>properties_by_type()</pre>		-		luato	or.workflow.Workflow	Graph
(openff.evaluator.datasets.thermoml.ThermoMLL		proper	•		( <b>1</b>	
method), 145	Protoco				(class	in
<pre>property_name (openff.evaluator.workflow.utils.ProtocolF</pre>		x			<pre>ckflow.schemas), 272 r.datasets.Calculatior</pre>	Source
property_phase (openff.evaluator.storage.data.BaseSimu				uaior		isource
attribute), 226				evalu	ator.protocols.paprik	ka.coordinates.PrepareP
<pre>property_phase (openff.evaluator.storage.data.StoredFre</pre>					I I I I I I I I I I I I I I I I I I I	I ·····
attribute), 234				oenff.	evaluator.protocols.p	paprika.coordinates.Prep
<pre>property_phase(openff.evaluator.storage.data.StoredSin attribute), 230</pre>	•					
$\verb property_phase (open ff. evaluator. storage. query. BaseSimple in the storage of the storage$	u <b>la</b> tionDa	ıtaQuery	v			
attribute), 240 property_phase (openff.evaluator.storage.query.FreeEne	query() rgyDataQ	o) Uniethod	penff.eva D. 218	luate	pr.storage.LocalFileS	torage
attribute), 246	query()	(	openff.ev	alua	tor.storage.StorageBa	ackend
$\verb property_phase (open \textit{ff.evaluator.storage.query.Simulations)  } \\$	onDataQu	ighethod	2), 216		0 0	
attribute), 244	QueryAt	tribut	e		(class	in
property_types (openff.evaluator.datasets.PhysicalProperty_types)						
property), 139	queued_	proper	ties(op	penff.	evaluator.client.Requ	iestResult
<pre>property_types (openff.evaluator.datasets.taproom.Tapro property), 150</pre>					r	4-1
property_types (openff.evaluator.datasets.thermoml.The	rmoMLDa	a <b>ta Sat</b> te)	), 91		f.evaluator.server.Bat	
property), 145 PropertyPhase (class in openff.evaluator.datasets), 96	QueueWo				(class	in
Protocol (class in openff.evaluator.workflow), 261	QuanaMa				kends), 205 UToolkit (class	in
protocol_ids (openff.evaluator.workflow.utils.ProtocolPa property), 284					UToolkit ( <i>class</i> <i>kends</i> ), 206	in
protocol_path (openff.evaluator.workflow.utils.ProtocolH	Path					
property), 285	recepto	r_coor	dinate_	fil	e	
protocol_replicators (openff.evaluator.workflow.schemas.WorkflowSch	ета		Eevaluato te), 342	or.pro	otocols.coordinates.B	BuildDockedCoordinates
attribute), 279	recepto	r_resi	due_nam	ne		
protocol_schemas(openff.evaluator.workflow.schemas.P	roiocoiGr	(openff	evaluato?	or.pro	otocols.coordinates.B	<i>BuildDockedCoordinates</i>

attribute), 343	method), 322
	replace_protocol() (openff.evaluator.protocols.coordinates.BuildCoord
(openff.evaluator.protocols.yank.LigandReceptor)	
	<pre>replace_protocol() (openff.evaluator.protocols.coordinates.BuildDocke</pre>
${\tt reference} \ (open {\it ff.evaluator.datasets.Measurement} Source$	<i>method</i> ), 346
attribute), 99	replace_protocol() (openff.evaluator.protocols.coordinates.SolvateExis
reference_reduced_potentials	method), 340
(openff.evaluator.protocols.reweighting.BaseMBA attribute), 501	<b>Reptace</b> [protocol() (openff.evaluator.protocols.forcefield.BaseBuildSyst method), 351
	replace_protocol() (openff.evaluator.protocols.forcefield.BuildLigParG
(openff.evaluator.protocols.reweighting.Reweight.	
	replace_protocol() (openff.evaluator.protocols.forcefield.BuildSmirnoff
reference_reduced_potentials	<i>method</i> ), 356
$(open {\it ff.evaluator.protocols.reweighting.Reweight)$	<b>Deptrackle</b> protocol() (openff.evaluator.protocols.forcefield.BuildTLeapSy
attribute), 510	<i>method</i> ), 367
	replace_protocol() (openff.evaluator.protocols.gradients.ZeroGradient:
<pre>openff.evaluator.layers), 193 register_calculation_schema() (in module</pre>	method), 372
register_calculation_schema() (in module openff.evaluator.layers), 193	<pre>replace_protocol() (openff.evaluator.protocols.groups.ConditionalGrou method), 379</pre>
	replace_protocol() (openff.evaluator.protocols.miscellaneous.AddValue
openff.evaluator.plugins), 572	method), 384
	replace_protocol() (openff.evaluator.protocols.miscellaneous.DivideVa
openff.evaluator.plugins), 572	<i>method</i> ), 398
	<pre>replace_protocol() (openff.evaluator.protocols.miscellaneous.DummyP</pre>
openff.evaluator.datasets.thermoml), 146	method), 413
	replace_protocol() (openff.evaluator.protocols.miscellaneous.FilterSub
openff.evaluator.workflow), 272	method), 409 a <b>tepdr&amp;depro</b> tocol() (openff.evaluator.protocols.miscellaneous.Multiply)
attribute), 191	method), 393
	<b>replaightingSthand</b> () (openff.evaluator.protocols.miscellaneous.Subtract)
attribute), 202	<i>method</i> ), 389
relative_tolerance(openff.evaluator.layers.simulation.	Siepllatien Schwarol () (openff.evaluator.protocols.miscellaneous.WeightBy
attribute), 199	<i>method</i> ), 403
	arkylbacealpubationshipmupenff.evaluator.protocols.openmm.OpenMMEnerg
attribute), 196	method), 419
attribute), 460	meplsaGenprostRelade()Respectives and the second states of the second se
	replace_protocol() (openff.evaluator.protocols.openmm.OpenMMSimu
method), 251	method), 425
	ApplageDiptort.cicCb())t(mtenff.evaluator.protocols.paprika.analysis.Analyz
<i>method</i> ), 302	<i>method</i> ), 473
	replage [prbrogol.() (openff.evaluator.protocols.paprika.analysis.Compu
method), 307	<i>method</i> ), 483
	<b>Applage</b> () <b>psertokla</b> () (openff.evaluator.protocols.paprika.analysis.Computed by 470
method), 296	method), 478 BapkAcergeObseolaDleopenff.evaluator.protocols.paprika.coordinates.Ad
method), 290	method), 447
	<b>BapkBee_predataePolio</b> Coppenff.evaluator.protocols.paprika.coordinates.Pre
method), 317	<i>method</i> ), 436
	MaphateDpokdon() (sopenff.evaluator.protocols.paprika.coordinates.Pre
<i>method</i> ), 312	<i>method</i> ), 441
	Paplaced_apaConstant & Copenff.evaluator.protocols.paprika.restraints.Apply
method), 327	method), 468
reprace_protocor() (openff.evaluator.protocols.analysis	<b>Paplaced_pafot_actd</b> ry) (openff.evaluator.protocols.paprika.restraints.Gene.

<i>method</i> ), 452		required	d_effective_samples
<pre>replace_protocol()</pre>	(openff.evaluator.protocols.paprika	.restraints.	GepeenfitesRuhhRostpaintecols.reweighting.ReweightDielectricConsta
<i>method</i> ), 457			attribute), 516
replace_protocol()	(openff.evaluator.protocols.paprika		
<i>method</i> ), 462			$(open {\it ff.evaluator.protocols.reweighting.} Reweight Observable$
replace_protocol()	(openff.evaluator.protocols.reweigh	-	
<i>method</i> ), 499			$\verb"d_inputs" (open {\it ff.evaluator.protocols.analysis.AverageDielectric Content of the state of$
replace_protocol()	(openff.evaluator.protocols.reweigh	-	
<i>method</i> ), 504			$\texttt{d_inputs} (\textit{openff.evaluator.protocols.analysis.AverageFreeEnerging} and \texttt{a} and \texttt{b} a$
	(openff.evaluator.protocols.reweigh	-	
<i>method</i> ), 493			d_inputs(openff.evaluator.protocols.analysis.AverageObservable
	(openff.evaluator.protocols.reweigh	-	
<i>method</i> ), 489	(		d_inputs (openff.evaluator.protocols.analysis.BaseAverageObser
	(openff.evaluator.protocols.reweigh	-	
<i>method</i> ), 516			d_inputs (openff.evaluator.protocols.analysis.BaseDecorrelatePr
	(openff.evaluator.protocols.reweigh	-	
<i>method</i> ), 510			d_inputs (openff.evaluator.protocols.analysis.ComputeDipoleMo
	(openff.evaluator.protocols.simulati		
<i>method</i> ), 521	(		d_inputs (openff.evaluator.protocols.analysis.DecorrelateObserv
	(openff.evaluator.protocols.simulati		
method), 528	( an auff auglu at an invational a standard		d_inputs (openff.evaluator.protocols.analysis.DecorrelateTraject
method), 534	(openff.evaluator.protocols.storage.	-	
, , , , , , , , , , , , , , , , , , ,	(openff.evaluator.protocols.yank.Ba		d_inputs (openff.evaluator.protocols.coordinates.BuildCoordinat
method), 540	(орепуј.е чанают.ргоюсок.уанк.ва		d_inputs (openff.evaluator.protocols.coordinates.BuildDockedCo
· · ·	(openff.evaluator.protocols.yank.Li		
method), 548	(openg).evaluator.protocots.yank.El		d_inputs (openff.evaluator.protocols.coordinates.SolvateExisting
· · ·	(openff.evaluator.protocols.yank.So		
<i>method</i> ), 556	( <i>cp cry</i> ), <i>c c c c c c c c c c</i>		d_inputs (openff.evaluator.protocols.forcefield.BaseBuildSystem)
	(openff.evaluator.workflow.Protoco		<i>property</i> ), 351
<i>method</i> ), 263			d_inputs(openff.evaluator.protocols.forcefield.BuildLigParGenS
	(openff.evaluator.workflow.Protoco		property), 362
<i>method</i> ), 269			d_inputs(openff.evaluator.protocols.forcefield.BuildSmirnoffSyst
<pre>replace_protocol()</pre>	(openff.evaluator.workflow.utils.Pro		
<i>method</i> ), 285		required	d_inputs(openff.evaluator.protocols.forcefield.BuildTLeapSystem
<pre>replace_protocol()</pre>	(openff.evaluator.workflow.Workflow	W	property), 367
<i>method</i> ), 255		required	d_inputs(openff.evaluator.protocols.gradients.ZeroGradients
replace_protocol_ty	ypes()		property), 373
(openff.evalua	tor.workflow.schemas.WorkflowSch	e <b>n</b> equired	d_inputs(openff.evaluator.protocols.groups.ConditionalGroup
<i>method</i> ), 279			property), 379
ReplaceableData		required	d_inputs(openff.evaluator.protocols.miscellaneous.AddValues
	or.storage.data), 224		property), 384
ReplicatorValue			d_inputs(openff.evaluator.protocols.miscellaneous.DivideValue
	or.workflow.utils), 283		property), 399
Request (class in open			d_inputs(openff.evaluator.protocols.miscellaneous.DummyProto
	(openff.evaluator.client.EvaluatorC		property), 414
<i>method</i> ), 79			d_inputs(openff.evaluator.protocols.miscellaneous.FilterSubstan
	valuator.forcefield.LigParGenForce		
property), 186			d_inputs (openff.evaluator.protocols.miscellaneous.MultiplyValu
	s in openff.evaluator.client), 84		property), 394
-	in openff.evaluator.client), 86		d_inputs (openff.evaluator.protocols.miscellaneous.SubtractValu
required_effective_			property), 389
			A_inputs (openff.evaluator.protocols.miscellaneous.WeightByMo           property)         404
attribute), 501			property), 404

required_inputs(openff.evaluator.protocols.openmm.OpmelfWIEe	edrzyckieniau stype ()
property), 419	(openff.evaluator.layers.CalculationLayer
required_inputs(openff.evaluator.protocols.openmm.OpenMMEv	valduaise limethyid 3, 189
property), 431 require	ed_schema_type()
required_inputs(openff.evaluator.protocols.openmm.OpenMMSi	m(dptioff.evaluator.layers.reweighting.ReweightingLayer
property), 425	class method), 200
required_inputs(openff.evaluator.protocols.paprika.analysiquAna	klyzstRenduatype()
property), 474	(openff.evaluator.layers.simulation.SimulationLayer
required_inputs(openff.evaluator.protocols.paprika.analysis.Com	
	ed_schema_type()
required_inputs (openff.evaluator.protocols.paprika.analysis.Com	
property), 479	class method), 194
required_inputs (openff.evaluator.protocols.paprika.coombistment	
property), 447	attribute), 452
required_inputs (openff.evaluator.protocols.paprika.coombistareai	
<i>property</i> ), 437	attribute), 458
required_inputs (openff.evaluator.protocols.paprika.coombisturati	
property), 442	attribute), 463
required_inputs (openff.evaluator.protocols.paprika.restmentstationa)	
property), 468	attribute), 544
required_inputs (openff.evaluator.protocols.paprika.restmestsr@i	
property), 452	attribute), 470
required_inputs (openff.evaluator.protocols.paprika.rest <b>mest</b> sr@i	
property), 458	attribute), 480
required_inputs (openff.evaluator.protocols.paprika.rest <b>mest</b> srGi	
property), 463	attribute), 465
required_inputs (openff.evaluator.protocols.reweighting. <b>BestFed</b>	
	attribute), 452
property), 499	
required_inputs (openff.evaluator.protocols.reweighting. <b>BestMan</b>	
property), 505	attribute), 458
required_inputs (openff.evaluator.protocols.reweighting.	
property), 494	attribute), 463
required_inputs ( <i>openff.evaluator.protocols.reweighting.</i>	
property), 489	attribute), 304
required_inputs(openff.evaluator.protocols.reweighting. <b>Resul</b> gh	
property), 516	attribute), 381
required_inputs(openff.evaluator.protocols.reweighting. <b>Resul</b> gh	
property), 510	attribute), 395
required_inputs(openff.evaluator.protocols.simulation.BresElter	
property), 522	attribute), 391
required_inputs(openff.evaluator.protocols.simulation.BresSibut	
property), 529	attribute), 386
required_inputs(openff.evaluator.protocols.storage.Unparelsudited	
property), 535	attribute), 470
required_inputs(openff.evaluator.protocols.yank.BaseYandsBuddo	
property), 541	attribute), 480
required_inputs(openff.evaluator.protocols.yank.LigandResulton	( <b>VpakJfrexaha</b> ltor.protocols.paprika.analysis.ComputeSymmetryCom
property), 548	attribute), 475
required_inputs(openff.evaluator.protocols.yank.SolvativeEvalkte	of appendix and the contract of the contract o
property), 557 retain_	_packmol_files
required_inputs (openff.evaluator.workflow.Protocol	$(open {\it ff.evaluator.protocols.coordinates.BuildCoordinatesPackmondspaces and the second s$
property), 262	attribute), 330
required_inputs(openff.evaluator.workflow.ProtocolGrompetain_	_packmol_files
property), 268	$(open {\it ff.evaluator.protocols.coordinates.Solvate Existing Structure} \\$

attribute), 340 (openff.evaluator.layers.simulation.Simulat	tion I aver
retrieve_force_field() (openji.evaluator.tayers.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.stmatation.st	ionLayer
(openff.evaluator.storage.LocalFileStorage schedule_calculation()	
method), 218 (openff.evaluator.layers.workflow.Workflow	CalculationLayer
retrieve_force_field() class method), 194	
(openff.evaluator.storage.StorageBackend schema (openff.evaluator.protocols.analysis.Average	DielectricConstant
method), 216 property), 302	
<pre>retrieve_object() (openff.evaluator.storage.LocalFileStandyema (openff.evaluator.protocols.analysis.Average method), 218</pre> property), 308	FreeEnergies
retrieve_object() (openff.evaluator.storage.StorageBacksathema (openff.evaluator.protocols.analysis.Average	Observable
method), 216 property), 296	
$\verb"retrieve_results()" (open ff. evaluator. client. Evaluator Client the matching of the constraint o$	rageObservable
method), 79 property), 291	
ReweightDielectricConstant (class in schema (openff.evaluator.protocols.analysis.BaseDec	correlateProtocol
openff.evaluator.protocols.reweighting), 511 property), 318 ReweightingLayer (class in schema (openff.evaluator.protocols.analysis.Compute	a Dipola Moments
openff.evaluator.layers.reweighting), 199 property), 313	eDipotemomenis
ReweightingProtocols (class in schema (openff.evaluator.protocols.analysis.Decorre	lateObservables
openff.evaluator.protocols.utils), 560 property), 328	
ReweightingSchema (class in schema (openff.evaluator.protocols.analysis.Decorre	lateTrajectory
openff.evaluator.layers.reweighting), 201 property), 323	
ReweightObservable         (class         in         schema (openff.evaluator.protocols.coordinates.Build)	dCoordinatesPackmol
openff.evaluator.protocols.reweighting), 505 property), 334	
rfind() (openff.evaluator.storage.attributes.FilePath schema (openff.evaluator.protocols.coordinates.Build method), 252 property), 346	dDockedCoordinates
right_hand_value(openff.evaluator.protocols.groups.Corstitienal@penff.@omkdiatiom.protocols.coordinates.Solva	ateExistingStructure
attribute), 375 property), 340	
$\texttt{rindex()}  (\textit{openff.evaluator.storage.attributes.FilePath}  \texttt{schema}(\textit{openff.evaluator.protocols.forcefield.BaseButcher}) \\ = \texttt{schema}(\texttt{openff.evaluator.protocols.forcefield.BaseButcher}) \\ = schem$	uildSystem
method), 252 property), 351	
rjust() (openff.evaluator.storage.attributes.FilePath schema (openff.evaluator.protocols.forcefield.BuildLu method), 252 property), 362	igParGenSystem
role (openff.evaluator.substances.Component attribute), schema (openff.evaluator.protocols.forcefield.BuildSt	mirnoffSystem
129 property), 356	nin nog system
$\verb"root_directory" (open ff. evaluator. storage. Local File Storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cefield. Build The storage chema (open ff. evaluator. protocols. for cef$	LeapSystem
property), 217 property), 367	
root_protocols (openff.evaluator.workflow.ProtocolGraphschema (openff.evaluator.protocols.gradients.ZeroGr	radients
property), 266 property), 373	
root_protocols (openff.evaluator.workflow.WorkflowGrapschema (openff.evaluator.protocols.groups.Condition property), 258 property), 379	laiGroup
<b>rpartition()</b> (openff.evaluator.storage.attributes.FilePathschema (openff.evaluator.protocols.miscellaneous.Ac	ldValues
<i>method</i> ), 252 <i>property</i> ), 384	
<pre>rsplit() (openff.evaluator.storage.attributes.FilePath schema (openff.evaluator.protocols.miscellaneous.Di method), 252 property), 399</pre>	ivideValue
rstrip() (openff.evaluator.storage.attributes.FilePath schema (openff.evaluator.protocols.miscellaneous.Di	ummyProtocol
method), 252 property), 414	-
schema (openff.evaluator.protocols.miscellaneous.Fi	lterSubstanceByRole
S property), 409	
schedule_calculation() schema(openff.evaluator.protocols.miscellaneous.Ma	ultiplyValue
(openff.evaluator.layers.CalculationLayer property), 394	htractValues
class method), 189 schema (openff.evaluator.protocols.miscellaneous.Su property), 389	unacivalues
schedule_calculation() property), 389 (openff.evaluator.layers.reweighting.ReweightingISfybema (openff.evaluator.protocols.miscellaneous.We	eightByMoleFraction
class method), 200 property), 404	5 ··· ,
schedule_calculation() schema(openff.evaluator.protocols.openmm.OpenMl	<b>MEnergyMinimisation</b>
···	

property), 419 openff.evaluator.datasets.curation.components.selec	tion),
schema (openff.evaluator.protocols.openmm.OpenMMEvaluateEnergie	
property), 431 SelectDataPointsSchema (class in	
schema (openff.evaluator.protocols.openmm.OpenMMSimulation property), 425 openff.evaluator.datasets.curation.components.select 176	tion),
schema (openff.evaluator.protocols.paprika.analysis.Analyz Stances (class in	
property), 474 openff.evaluator.datasets.curation.components.selec	tion),
${\tt schema}\ (open ff. evaluator. protocols. paprika. analysis. Compute Reference Work$	
property), 484 SelectSubstancesSchema (class in	
schema (openff.evaluator.protocols.paprika.analysis.ComputeSymmetryCenffkevtibutor.datasets.curation.components.selec property), 479 175	tion),
${\tt schema} \ (open {\it ff.evaluator.protocols.paprika.coordinates.} Add {\tt Parmery} {\tt Arddasess} \ (open {\it ff.evaluator.client.ConnectionOptions} \ {\tt open ff.evaluator.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.client.clien$	
property), 447 attribute), 81	
schema (openff.evaluator.protocols.paprika.coordinates.Pre <b>para?end]6ddm/isst</b> (aspenff.evaluator.client.EvaluatorClient	
property), 437 property), 78	
schema (openff.evaluator.protocols.paprika.coordinates.PreparaRalepsaCt@mplinfftesaluator.client.ConnectionOptions property), 442 attribute), 81	
schema (openff.evaluator.protocols.paprika.restraints.Apply <b>Bervia</b> intport (openff.evaluator.client.EvaluatorClient	
property), 468 property), 78	
schema (openff.evaluator.protocols.paprika.restraints.Genersut Auuakt (strainenff.evaluator.protocols.analysis.AverageDiet	ectricConstar
property), 453 method), 302	
schema (openff.evaluator.protocols.paprika.restraints.GenersatPullRet(f)dimpenff.evaluator.protocols.analysis.AverageFree	Energies
property), 458 method), 308	
schema (openff.evaluator.protocols.paprika.restraints.GenersataRehaisABe(sopeinffsevaluator.protocols.analysis.AverageObs property), 463 method), 296	ervable
schema (openff.evaluator.protocols.reweighting.BaseEvaluater.protocols.analysis.BaseAverag property), 499 method), 291	2Observable
schema (openff.evaluator.protocols.reweighting.BaseMBARBettoutld() (openff.evaluator.protocols.analysis.BaseDecorr	elateProtocol
property), 505 method), 318	
schema (openff.evaluator.protocols.reweighting.Concatenate@bsavvidl@s(openff.evaluator.protocols.analysis.ComputeDip property), 494 method), 313	oleMoments
schema (openff.evaluator.protocols.reweighting.Concatenat SFrujawick) (openff.evaluator.protocols.analysis.Decorrelate	Observables
property), 489 method), 328	000000
${\tt schema} (open ff. evaluator. protocols. reweighting. Reweight Disket triax in the set of the s$	Trajectory
property), 516 method), 323	
schema (openff.evaluator.protocols.reweighting.ReweightObsetvahlad() (openff.evaluator.protocols.coordinates.BuildCo	ordinatesPack
property), 510 method), 334	
schema (openff.evaluator.protocols.simulation.BaseEnergyMintunition) (openff.evaluator.protocols.coordinates.BuildDo	ckedCoordind
property), 522 method), 346 schema (openff.evaluator.protocols.simulation.BaseSimulatiset_uuid() (openff.evaluator.protocols.coordinates.SolvateE	wisting Structs
property), 529 method), 340	xistingStructu
schema (openff.evaluator.protocols.storage.UnpackStoredSimetationild(i)a(openff.evaluator.protocols.forcefield.BaseBuild	System
property), 535 method), 351	ystem
schema (openff.evaluator.protocols.yank.BaseYankProtocol set_uuid() (openff.evaluator.protocols.forcefield.BuildLigPa	urGenSystem
property), 541 method), 362	2
schema (openff.evaluator.protocols.yank.LigandReceptorYansleftrouvidl() (openff.evaluator.protocols.forcefield.BuildSmirr property), 548 method), 356	offSystem
$schema (\textit{openff.evaluator.protocols.yank.SolvationYankProtect_uuid()} (\textit{openff.evaluator.protocols.forcefield.BuildTLeaplacet_uuid()}) (openff.evaluator.protocols.forcefield.BuildTLeaplacet_uu$	oSystem
schema (openff.evaluator.workflow.Protocol property), set_uuid() (openff.evaluator.protocols.gradients.ZeroGradi	ents
262 method), 373 schema (openff.evaluator.workflow.ProtocolGroup prop-set_uuid() (openff.evaluator.protocols.groups.ConditionalC	Froup
<i>erty</i> ), 271 <i>method</i> ), 379	,
SelectDataPoints       (class       in       set_uuid() (openff.evaluator.protocols.miscellaneous.AddVa	ılues

<i>method</i> ), 384	<i>method</i> ), 529
<pre>set_uuid() (openff.evaluator.protocols.miscellaneous.DiviseValue</pre>	
<i>method</i> ), 399	<i>method</i> ), 535
<pre>set_uuid() (openff.evaluator.protocols.miscellaneous.DunsetPunt method), 414</pre>	bd() (openff.evaluator.protocols.yank.BaseYankProtocol method), 541
set_uuid() (openff.evaluator.protocols.miscellaneous.Filtes Stubsman	
method), 409	method), 548
set_uuid() (openff.evaluator.protocols.miscellaneous.MulsetyValla	
method), 394	method), 557
set_uuid() (openff.evaluator.protocols.miscellaneous.Subtsect_Valuation	ids() (openff.evaluator.workflow.Protocol
method), 389	<i>method</i> ), 263
set_uuid() (openff.evaluator.protocols.miscellaneous.WeigslerByWhat	ldeFractupenff.evaluator.workflow.ProtocolGroup
<i>method</i> ), 404	<i>method</i> ), 268
<pre>set_uuid() (openff.evaluator.protocols.openmm.OpenMMEetrgyd</pre>	<b>Liui: hi) s(atpem</b> ff.evaluator.protocols.analysis.AverageDielectricConsta
<i>method</i> ), 419	<i>method</i> ), 302
<pre>set_uuid() (openff.evaluator.protocols.openmm.OpenMMEexlumate</pre>	<b>Ere</b> ( <i>glepenff.evaluator.protocols.analysis.AverageFreeEnergies</i>
<i>method</i> ), 431	method), 308
<pre>set_uuid() (openff.evaluator.protocols.openmm.OpenMMSierulatil method), 426</pre>	<b>bae</b> () (openff.evaluator.protocols.analysis.AverageObservable method), 296
set_uuid() (openff.evaluator.protocols.paprika.analysis.AselyzeAl	
method), 474	method), 291
set_uuid() (openff.evaluator.protocols.paprika.analysis.Computer	lafe (@) (@) Veaff evaluator.protocols.analysis.BaseDecorrelateProtoco
method), 484	method), 318
<pre>set_uuid() (openff.evaluator.protocols.paprika.analysis.ComputedS</pre>	www.figthoyfeonffectionator.protocols.analysis.ComputeDipoleMoment.
<i>method</i> ), 479	<i>method</i> ), 313
set_uuid()(openff.evaluator.protocols.paprika.coordinatesetdddal	and Mathematics. A state of the second state o
<i>method</i> ), 447	method), 328
<pre>set_uuid() (openff.evaluator.protocols.paprika.coordinatesdfr_pail</pre>	aRe(I)Copedificated uctor.protocols.analysis.DecorrelateTrajectory
<i>method</i> ), 437	<i>method</i> ), 323
<pre>set_uuid() (openff.evaluator.protocols.paprika.coordinatesaPr_pal</pre>	
<i>method</i> ), 442	<i>method</i> ), 334
<pre>set_uuid() (openff.evaluator.protocols.paprika.restraints.AspplyRad</pre>	
<i>method</i> ), 468	<i>method</i> ), 346
<pre>set_uuid() (openff.evaluator.protocols.paprika.restraints.Generate</pre>	
<i>method</i> ), 453	method), 340
set_uuid() (openff.evaluator.protocols.paprika.restraints.Generat	
method), 458	method), 351
set_uuid() (openff.evaluator.protocols.paprika.restraints.Generat	
method), 463	method), 362
<pre>set_uuid() (openff.evaluator.protocols.reweighting.BaseEschuater method), 499</pre>	method), 356
set_uuid() (openff.evaluator.protocols.reweighting.BaseMBATRAR	
method), 505	method), 368
set_uuid() (openff.evaluator.protocols.reweighting.Concasetuteal	
method), 494	method), 373
set_uuid() (openff.evaluator.protocols.reweighting.Concasetute/Fr	
method), 489	method), 379
set_uuid() (openff.evaluator.protocols.reweighting.Reweight	
<i>method</i> ), 516	method), 384
set_uuid() (openff.evaluator.protocols.reweighting.Reweight	
<i>method</i> ), 510	method), 399
set_uuid() (openff.evaluator.protocols.simulation.BaseEnsegyMind	
method), 522	method), 414
<pre>set_uuid() (openff.evaluator.protocols.simulation.BaseSinsetution)</pre>	Lue() (openff.evaluator.protocols.miscellaneous.FilterSubstanceBy

	nethod), 548
set_value() (openff.evaluator.protocols.miscellaneous.Mu <b>stiplyWallue</b>	
	nethod), 557
<pre>set_value() (openff.evaluator.protocols.miscellaneous.SubsetictWalling</pre>	
	nethod), 264
<pre>set_value() (openff.evaluator.protocols.miscellaneous.WeseftBydMade method), 404</pre>	<b>eF</b> a(ajumff.evaluator.workflow.ProtocolGroup nethod), 271
<pre>set_value() (openff.evaluator.protocols.openmm.OpenMMsEndegfamil</pre>	<b>ትተሱ៌)\$‹atpen</b> ff.evaluator.utils.observables.ObservableFrame nethod), 570
set_value() (openff.evaluator.protocols.openmm.OpenMMsEtunpagerE	
set_value() (openff.evaluator.protocols.openmm.OpenMMs&inuplacin	
	uttribute), 548
<pre>set_value() (openff.evaluator.protocols.paprika.analysis.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.seal sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sealype.sea</pre>	RP (mpsenff.evaluator.protocols.yank.SolvationYankProtocol
	uttribute), 557
set_value() (openff.evaluator.protocols.paprika.analysis.&mplatRe	
	$open ff. evaluator. protocols. storage. Unpack {\it StoredSimulationData}$
<pre>set_value() (openff.evaluator.protocols.paprika.analysis.ComputeSp</pre>	
	onDataQuery (class in
<pre>set_value() (openff.evaluator.protocols.paprika.coordinates.AddDua</pre>	
method), 447 Simulatio	-
<pre>set_value() (openff.evaluator.protocols.paprika.coordinates.Prepare</pre>	
	onProtocols (class in
<pre>set_value() (openff.evaluator.protocols.paprika.coordinates.Prepare</pre>	
method), 442 Simulatio	
<pre>set_value() (openff.evaluator.protocols.paprika.restraints.ApplyRest</pre>	
	(openff.evaluator.substances.Component at-
<pre>set_value() (openff.evaluator.protocols.paprika.restraints.Generateff</pre>	
	ForceFieldSource (class in
<pre>set_value() (openff.evaluator.protocols.paprika.restraints.Generateb</pre>	
	penff.evaluator.protocols.yank.SolvationYankProtocol
set_value() (openff.evaluator.protocols.paprika.restraints.Generate	
	oordinate_file
<pre>set_value() (openff.evaluator.protocols.reweighting.BaseEvaluateEn</pre>	
method), 499 set_value() (openff.evaluator.protocols.reweighting.Base MBAR Promo	attribute), 336
method), 505 (a set_value() (openff.evaluator.protocols.reweighting.ConcatenateOba	openff.evaluator.protocols.yank.SolvationYankProtocol
	_1_free_energy
set_value() (openff.evaluator.protocols.reweighting.ConcatenateTrag	
<i>method</i> ), 489 a	attribute), 552
	attribute), 552
set_value() (openff.evaluator.protocols.reweighting.Rewe <b>ightOdison</b> y	<u>vábl</u> ærajectory_path
method), 510 (d	openff.evaluator.protocols.yank.SolvationYankProtocol
<pre>set_value() (openff.evaluator.protocols.simulation.BaseEnergyMinia</pre>	untisittudei), 553
	_2_coordinates
<pre>set_value() (openff.evaluator.protocols.simulation.BaseSimulation (</pre>	
	uttribute), 552
<pre>set_value() (openff.evaluator.protocols.storage.UnpackStstellStimute</pre>	
	openff.evaluator.protocols.yank.SolvationYankProtocol
	uttribute), 553
	_2_system(openff.evaluator.protocols.yank.SolvationYankProto
<pre>set_value() (openff.evaluator.protocols.yank.LigandReceptorYankPa</pre>	uoticiconte), 332

source (*openff.evaluator.properties.ExcessMolarVolume* solution_2_trajectory_path (openff.evaluator.protocols.vank.SolvationYankProtocol attribute). 107 source (openff.evaluator.properties.HostGuestBindingAffinity attribute), 553 solvated_complex_coordinates attribute), 124 (openff.evaluator.protocols.yank.LigandReceptorYsokPathopenff.evaluator.properties.SolvationFreeEnergy attribute), 544 attribute), 120 solvated_complex_system source calculation id (openff.evaluator.protocols.yank.LigandReceptorYankProtoc@penff.evaluator.storage.data.BaseSimulationDataattribute). 544 attribute), 226 solvated_complex_trajectory_path source_calculation_id (openff.evaluator.protocols.yank.LigandReceptorYankProtoc@penff.evaluator.storage.data.StoredFreeEnergyDataattribute), 545 attribute), 234 solvated_ligand_coordinates source_calculation_id (openff.evaluator.protocols.yank.LigandReceptorYankProtocolpenff.evaluator.storage.data.StoredSimulationDataattribute), 543 attribute), 230 solvated_ligand_system source_calculation_id (openff.evaluator.protocols.yank.LigandReceptorYankProtoc@penff.evaluator.storage.query.BaseSimulationDataQuery attribute), 543 attribute), 240 solvated_ligand_trajectory_path source_calculation_id (openff.evaluator.protocols.yank.LigandReceptorYankProtocolpenff.evaluator.storage.query.FreeEnergyDataQuery attribute), 544 attribute), 246 SolvateExistingStructure (class in source_calculation_id openff.evaluator.protocols.coordinates), 335 (openff.evaluator.storage.query.SimulationDataQuery SolvationFreeEnergy (class attribute), 244 in openff.evaluator.properties), 118 sources (openff.evaluator.datasets.PhysicalPropertyDataSet SolvationYankProtocol (class in property), 139 openff.evaluator.protocols.yank), 549 sources (openff.evaluator.datasets.taproom.TaproomDataSet solvent_1(openff.evaluator.protocols.yank.SolvationYankProtocol property), 150 attribute), 551 sources (openff.evaluator.datasets.thermoml.ThermoMLDataSet solvent_1_coordinate_path property), 145 (openff.evaluator.protocols.yank.SolvationYankProsphilt() (openff.evaluator.storage.attributes.FilePath attribute), 552 method), 252 solvent_1_trajectory_path splitlines() (openff.evaluator.storage.attributes.FilePath (openff.evaluator.protocols.yank.SolvationYankProtocol method), 252 start() (openff.evaluator.backends.CalculationBackend attribute), 552 solvent_2 (openff.evaluator, protocols, yank. Solvation Yank Protocol method), 204 attribute), 551 start() (openff.evaluator.backends.dask.BaseDaskBackend solvent_2_coordinate_path method), 208 (openff.evaluator.protocols.yank.SolvationYankProstoend () (openff.evaluator.backends.dask.BaseDaskJobQueueBackend attribute), 553 method), 209 solvent_2_trajectory_path start() (openff.evaluator.backends.dask.DaskLocalCluster (openff.evaluator.protocols.yank.SolvationYankProtocol method), 210 attribute), 553 start() (openff.evaluator.backends.dask.DaskLSFBackend Source (class in openff.evaluator.datasets), 97 method), 212 source (openff.evaluator.datasets.PhysicalProperty atstart() (openff.evaluator.backends.dask.DaskPBSBackend tribute), 95 method), 214 source (openff.evaluator.properties.Density attribute), (openff.evaluator.server.EvaluatorServer start() 103 method), 90 **source** (*openff.evaluator.properties.DielectricConstant* start_protocol(openff.evaluator.workflow.utils.ProtocolPath attribute), 110 property), 284 (openff.evaluator.properties.EnthalpyOfMixing start_state_trajectory source attribute), 114 (openff.evaluator.storage.data.StoredFreeEnergyData source (openff.evaluator.properties.EnthalpyOfVaporization attribute), 232 attribute), 117 started (openff.evaluator.backends.CalculationBackend

	() 204				ar 1 (	(1) (1) (252)	
	property), 204		1	<i>a</i>	openff.evaluator.storag		015
started		ackends.dask.BaseDaskBa				<i>f.evaluator.storage</i> ),	215
	property), 208				orce_field()		
started		ackends.dask.BaseDaskJob	Que	eueBacker		ge.LocalFileStorage	
	property), 210			_	<i>method</i> ), 218		
started		ackends.dask.DaskLocalCl	uste	store_f		~ ~	
	property), 210		_	_	(openff.evaluator.storag	ge.StorageBackend	
started		ickends.dask.DaskLSFBac			<i>method</i> ), 215		_
	property), 212				bject()( <i>openff.evalua</i>	tor.storage.LocalFile	Storage
started		ackends.dask.DaskPBSBac			<i>method</i> ), 218		
	property), 214				bject()( <i>openff.evalua</i>	tor.storage.StorageB	ackend
startsw	ith()( <i>openff.evalu</i>	ator.storage.attributes.File			<i>method</i> ), 215		
	<i>method</i> ), 252				reeEnergyData	(class	in
State(c	lass in openff.evalua	tor.datasets.curation.comp				e.data), 231	
	177			StoredS	imulationData	(class	in
statist	ical_inefficien	су			openff.evaluator.storag	e.data), 228	
	(openff.evaluator.st	orage.data.StoredSimulati	onD	attrip()	(openff.evaluator.std	orage.attributes.FileI	Path
	attribute), 229				<i>method</i> ), 252		
steps_p	er_iteration			submit_	task() (openff.evaluate	or.backends.Calculat	ionBackend
	(openff.evaluator.pl	rotocols.openmm.OpenMM	1Sim	ulation	<i>method</i> ), 204		
	attribute), 426				task() (openff.evaluate	or.backends.dask.Bas	eDaskBackend
steps_r	er_iteration				method), 208		
		rotocols.simulation.BaseSi	mule	<i>s</i> ubmit_	task() (openff.evaluate	or.backends.dask.Bas	eDaskJobQueueBack
	attribute), 524				method), 209		~
steps r	er_iteration			submit	task() (openff.evaluate	or.backends.dask.Das	skLocalCluster
		rotocols.yank.BaseYankPro			<i>method</i> ), 210		
	attribute), 537				task() (openff.evaluate	or.backends.dask.Das	skLSFBackend
steps r	er_iteration				<i>method</i> ), 212		
		rotocols.yank.LigandRecep	otor}	subRicto	, .	or.backends.dask.Das	skPBSBackend
	attribute), 549				<i>method</i> ), 214		
stens r	er_iteration			subset(	) (openff.evaluator.utils.	observables.Observa	ableArray
ocopo_p		rotocols.yank.SolvationYan			method), 568		
	attribute), 557				) (openff.evaluator.utils.	observables Observa	ableFrame
steric		evaluator.protocols.yank.S				.00501100105.0050110	ioter runte
Sterre_	attribute), 552	evaluator.protocots.yank.s			ce (class in openff.evali	uator substances) 12	5
steric		evaluator.protocols.yank.S					
Sterre_	attribute), 552	evaluator.protocots.yank.s	oiva	Jundara	attribute), 94	uascis.i nysicaii rop	criy
ston()		ackends.CalculationBacker	nd	substan		properties Density	at-
stop()	(openg).evaluator.oc method), 204	ickenus.CuiculailonDucker	ш	Substan	tribute), 103	properties.Density	<i>ui</i> -
ston()(		ckends.dask.BaseDaskBack	and	cubetan		nartias DialactricCo	nstant
stop()(	method), 208	Kenus.uusk.buseDuskbuck	lenu	Substan	attribute), 110	periles.DielectricCo.	nstunt
ston()(		ckends.dask.BaseDaskJobQ	Juan	Bhotton		nartias Enthalm Of	livina
scop()(		kenas.aask.BaseDaskJOD	jueu	BDUSKIAN		pernes.EmmapyOjM	lixing
at an () (	<i>method</i> ), 210	akan da daak Daaki oo alChu		auh at an	<i>attribute</i> ), 114	monting Easth alm. Of	an ouis ation
stop()(		ckends.dask.DaskLocalClu	sier	Substan		periles.EninalpyOjV	aporization
	method), 210		,		attribute), 117		¥7 J
stop()(	1 00	ckends.dask.DaskLSFBack	end	substan	1 00 1	perfies.ExcessMolar	Volume
~	<i>method</i> ), 212				attribute), 107		1
stop()(		ckends.dask.DaskPBSBack	end	substan		perties.HostGuestBu	ndingAffinity
	<i>method</i> ), 214			_	attribute), 124		_
stop()		uator.server.EvaluatorServ	er	substan		perties.SolvationFree	eEnergy
	method), 90		-		attribute), 120		
storage		valuator.layers.reweighting	g.Re	seightiag		tocols.coordinates.B	uildCoordinatesPacki
	attribute), 201			_	attribute), 330	_	
Storage	Attribute	(class	in	substan	ce(openff.evaluator.pro	tocols.coordinates.So	olvateExistingStructu

attribute) 240	target	reduced notentials
attribute), 340 substance(openff.evaluator.protocols.forcefield.BaseBui		reduced_potentials (openff.evaluator.protocols.reweighting.BaseMBARProtocol
attribute), 348	iiusysiem	attribute), 501
substance (openff.evaluator.protocols.forcefield.BuildLig	g Patacifare Sty	
attribute), 362		(openff.evaluator.protocols.reweighting.ReweightDielectricConsta
substance (openff.evaluator.protocols.forcefield.BuildSm	nirnoffSyste	nattribute), 516
attribute), 356	target_	reduced_potentials
substance (openff.evaluator.protocols.forcefield.BuildTL attribute), 368	LeapSystem	(openff.evaluator.protocols.reweighting.ReweightObservable attribute), 510
${\tt substance} \ (open {\it ff.evaluator.protocols.paprika.coordinat} \\$	tes. <b>Facktfen</b> ts	
attribute), 444		$open {\it ff.evaluator.datasets.curation.components.selection}),$
substance (openff.evaluator.protocols.paprika.coordinat		
attribute), 437		ture (openff.evaluator.thermodynamics.ThermodynamicState
substance(openff.evaluator.protocols.paprika.coordinat	-	
attribute), 442 substance(openff.evaluator.protocols.storage.UnpackSto		ture_cutoff(openff.evaluator.layers.reweighting.ReweightingSc
attribute), 531		lynamic_state
substance (openff.evaluator.storage.data.BaseSimulation		(openff.evaluator.datasets.PhysicalProperty
attribute), 226	nDuiu	attribute), 94
substance (openff.evaluator.storage.data.StoredFreeEne	erg <b>thet</b> mod	
attribute), 234	0.	(openff.evaluator.properties.Density attribute),
substance (openff.evaluator.storage.data.StoredSimulation	ionData	103
attribute), 231		lynamic_state
<pre>substance(openff.evaluator.storage.query.BaseSimulatio</pre>	onDataQue	r(openff.evaluator.properties.DielectricConstant attribute), 110
substance (openff.evaluator.storage.query.FreeEnergyDe	atatennoc	
attribute), 246	_	(openff.evaluator.properties.EnthalpyOfMixing
substance (openff.evaluator.storage.query.SimulationDa		attribute), 114
attribute), 244		lynamic_state
substance_query (openff.evaluator.storage.query.BaseS attribute), 240		attribute), 117
substance_query (openff.evaluator.storage.query.FreeE attribute), 247		$(open {\it ff.evaluator.properties. Excess Molar Volume} \\$
substance_query (openff.evaluator.storage.query.Simul		
attribute), 244		lynamic_state
SubstanceQuery (class in	!	(openff.evaluator.properties.HostGuestBindingAffinity
openff.evaluator.storage.query), 236 substances (openff.evaluator.datasets.PhysicalProperty)	Datha	attribute), 124
property), 139	Ducascinoc	(openff.evaluator.properties.SolvationFreeEnergy
substances (openff.evaluator.datasets.taproom.Taproom	1DataSet	attribute), 120
property), 150		lynamic_state
		t (openff.evaluator.protocols.analysis.AverageDielectricConstant attribute), 302
SubtractValues (class in	thermod	lynamic_state
openff.evaluator.protocols.miscellaneous), 385		(openff.evaluator.protocols.analysis.AverageFreeEnergies attribute), 304
<pre>swapcase() (openff.evaluator.storage.attributes.FilePath</pre>	<i>i</i> thermod	lynamic_state
<i>method</i> ), 253		(openff.evaluator.protocols.analysis.AverageObservable attribute), 296
Т	thermod	lynamic_state
TaproomDataSet (class in		(openff.evaluator.protocols.analysis.BaseAverageObservable
openff.evaluator.datasets.taproom), 147		attribute), 287
TaproomSource (class in	thermod	lynamic_state
openff.evaluator.datasets.taproom), 151		(open ff. evaluator. protocols. open mm. Open MME valuate Energies

attribute), 431		.1	openff.evaluator.datasets.thermom	<i>l</i> ), 142
thermodynamic_state			stat_friction	
	protocols.openmm.OpenMM	usimulation		n.Openminisimulation
attribute), 426		+ h o rom o	attribute), 426	
thermodynamic_state	uotooola nanuika analuaia		stat_friction	tion Bass Cinculation
attribute), 480	rotocois.paprika.analysis.	ComputeRef	ere <b>(upeMfrk</b> valuator.protocols.simulat attribute), 525	ion.BaseSimulation
thermodynamic_state			eries_statistics	
(openff.evaluator.p attribute), 475	orotocols.paprika.analysis.	ComputeSyn	m <b>&amp;tpyGuff:eextlium</b> tor.protocols.analysi attribute), 302	s.AverageDielectricConstant
thermodynamic_state		time_s	eries_statistics	
(openff.evaluator.p	protocols.reweighting.Base	EvaluateEne	rgiæpenff.evaluator.protocols.analysi	s.AverageObservable
attribute), 496			attribute), 297	
thermodynamic_state			eries_statistics	
(openff.evaluator.p attribute), 512	protocols.reweighting.Rewe	eightDielectr	ic Copstuffie valuator.protocols.analysi attribute), 288	s.BaseAverageObservable
thermodynamic_state		time_s	eries_statistics	
(openff.evaluator.p attribute), 525	protocols.simulation.BaseS	imulation	(openff.evaluator.protocols.analysi attribute), 314	s.BaseDecorrelateProtocol
thermodynamic_state		time_s	eries_statistics	
	protocols.storage.UnpackSt		on <b>[Dptm</b> ]f.evaluator.protocols.analysi	s.DecorrelateObservables
attribute), 531			attribute), 328	
thermodynamic_state		time_s	eries_statistics	
(openff.evaluator.p attribute), 537	protocols.yank.BaseYankPr	rotocol	(openff.evaluator.protocols.analysi attribute), 323	s.DecorrelateTrajectory
thermodynamic_state		timest	ep (openff.evaluator.protocols.openm	nm.OpenMMSimulation
(openff.evaluator.p	protocols.yank.LigandRece	ptorYankPro	tocattribute), 426	-
attribute), 549		timest	ep (openff.evaluator.protocols.simula	ition.BaseSimulation
thermodynamic_state	motocols work Soluction Va	nh Duttimakt	<i>attribute</i> ), 524	DagaVanhBrotopol
(openj).evalualor.p attribute), 557	rolocols.yank.solvallon1a	NKP YOU DUDIE IS U	ep (openff.evaluator.protocols.yank.E attribute), 537	Sase Tank Protocol
thermodynamic_state		timoct	ep (openff.evaluator.protocols.yank.L	in an dPagantan Van LProto and
	torage.data.BaseSimulatio		attribute), 549	лдана Кесеріот Танк Гтоюсої
attribute), 226	iorage.aaia.basesimaiano		ep (openff.evaluator.protocols.yank.S	Solvation Vank Protocol
thermodynamic_state		LIMES	attribute), 557	οιναποπτατκή τοιοςοι
	torage.data.StoredFreeEne	erovDntritat]e(		utes FilePath
attribute), 234	iorage.aana.storear reeLine	si gybaile i C	method), 253	
thermodynamic_state		TLeap	orceFieldSource (class	in
-	torage.data.StoredSimulat	-	openff.evaluator.forcefield), 183	
attribute), 231			ce_field() (openff.evaluator.forcef	ield.SmirnoffForceFieldSourc
thermodynamic_state		00_101	method), 182	
-	torage.query.BaseSimulati	ionDat <b>oOme</b> t	· · ·	ow.Workflow
attribute), 240	0.1.5	20	<i>method</i> ), 256	<b>5</b>
thermodynamic_state		to_num	ber_of_molecules()	
(openff.evaluator.s	torage.query.FreeEnergyD		(openff.evaluator.substances.Amou	nt method),
attribute), 247			131	
thermodynamic_state		to_num	<pre>ber_of_molecules()</pre>	
(openff.evaluator.s	torage.query.SimulationDe	ataQuery	(openff.evaluator.substances.Exact	Amount
attribute), 244			method), 133	
ThermodynamicState	(class	<i>in</i> to_num	<pre>ber_of_molecules()</pre>	
1 00	ermodynamics), 136		(openff.evaluator.substances.Mole)	Fraction
<pre>thermoml_property()</pre>	(in modi		method), 135	
	atasets.thermoml), 146	to_par	das() (openff.evaluator.datasets.Phy	vsicalPropertyDataSet
ThermoMLDataSet	(class	in	<i>method</i> ), 140	

to_pandas() (openff.evaluator.datasets.taproom.Taprooml	
	trajectory_file_path
	MLDataSe(openff.evaluator.protocols.simulation.BaseSimulation
method), 145	attribute), 525
<pre>to_protocol() (openff.evaluator.workflow.schemas.Protoc</pre>	
<i>method</i> ), 275	(open ff. evaluator. protocols. storage. Unpack Stored Simulation Data and the store of the st
<pre>to_protocol() (openff.evaluator.workflow.schemas.Protoc</pre>	colSchemaattribute), 531
<i>method</i> ), 273	<pre>trajectory_path(openff.evaluator.protocols.analysis.ComputeDipoleMage)</pre>
to_storage_query() ( <i>openff.evaluator.storage.data.Base</i> .	Simulation <b>Dutib</b> ute), 309
<i>method</i> ), 227	trajectory_paths (openff.evaluator.protocols.paprika.analysis.AnalyzeA
to_storage_query() ( <i>openff.evaluator.storage.data.Base</i>	StoredDatattribute), 470
	translate() (openff.evaluator.storage.attributes.FilePath
to_storage_query() (openff.evaluator.storage.data.Force	
	type (openff.evaluator.protocols.groups.ConditionalGroup.Condition
to_storage_query() (openff.evaluator.storage.data.Hash	
	type (openff.evaluator.workflow.schemas.ProtocolGroupSchema
to_storage_query() ( <i>openff.evaluator.storage.data.Repla</i>	
	type (openff.evaluator.workflow.schemas.ProtocolSchema
to_storage_query() (openff.evaluator.storage.data.Store	
method), 233	ur reelinengy limmule), 275
to_storage_query() (openff.evaluator.storage.data.Stored	dimulation Data
method), 230	uncertainty (openff.evaluator.datasets.PhysicalProperty
tolerance (openff.evaluator.protocols.coordinates.BuildCo	· · · · · · · · · · · · · · · · · · ·
attribute), 330	uncertainty (openff.evaluator.properties.Density
tolerance (openff.evaluator.protocols.coordinates.Solvate)	ExistingStructfubute), 103
attribute), 340	<pre>uncertainty(openff.evaluator.properties.DielectricConstant</pre>
tolerance(openff.evaluator.protocols.openmm.OpenMME	nergyMinimisteliae), 110
attribute), 419	uncertainty (openff.evaluator.properties.EnthalpyOfMixing
$\verb+tolerance(openff.evaluator.protocols.simulation.BaseEne$	rgyMinimisatipute), 114
attribute), 518	uncertainty (openff.evaluator.properties.EnthalpyOfVaporization
topology_file_name( <i>openff.evaluator.storage.data.Store</i>	dFreeEnergy Aptae, 117
	uncertainty (openff.evaluator.properties.ExcessMolarVolume
topology_path(openff.evaluator.protocols.paprika.analys	is.AnalyzeAPRPhase107
	uncertainty (openff.evaluator.properties.HostGuestBindingAffinity
total_number_of_iterations	attribute), 124
	uktertainty (openff.evaluator.properties.SolvationFreeEnergy
attribute), 426	attribute), 120
(openff.evaluator.protocols.simulation.BaseSimula	UNDEFINED (in module openff.evaluator.attributes), 566
attribute), 524	
	openff.evaluator.protocols.storage), 529
	unsuccessful_properties
(openff.evaluator.protocols.storage.UnpackStored	
attribute), 531	tribute), 86
trajectory_file_name	unsuccessful_properties
(openff.evaluator.storage.data.StoredSimulationDe	ata (openff.evaluator.server.Batch attribute),
attribute), 229	91
trajectory_file_path	<pre>update() (openff.evaluator.utils.observables.ObservableFrame</pre>
$(open {\it ff.evaluator.protocols.open mm.Open MME valuator.protocols.open mm.Open mm.Open MME valuator.protocols.open mm.Open mm.$	luateEnersiethod), 570
	update_references()
trajectory_file_path	(openff.evaluator.workflow.schemas.ProtocolReplicator
(openff.evaluator.protocols.openmm.OpenMMSimedian and the set of	ulation method), 277
attribute), 426	upper() (openff.evaluator.storage.attributes.FilePath
trajectory_file_path	method), 253
(openff.evaluator.protocols.reweighting.BaseEvalu	

## **OpenFF Evaluator Documentation**

V	<pre>validate() (openff.evaluator.protocols.analysis.DecorrelateObservables</pre>
<pre>validate() (openff.evaluator.attributes.AttributeClass</pre>	method), 328
<i>method</i> ), 565	<pre>validate() (openff.evaluator.protocols.analysis.DecorrelateTrajectory method), 323</pre>
validate() (openff.evaluator.client.ConnectionOptions	validate() (openff.evaluator.protocols.coordinates.BuildCoordinatesPack
<pre>method), 81 validate() (openff.evaluator.client.Request method), 83</pre>	method), 334
validate() (openff.evaluator.client.RequestOptions	$\verb validate()  (open ff. evaluator. protocols. coordinates. Build Docked Coordinates. Build Build$
method), 85	method), 346
<pre>validate() (openff.evaluator.client.RequestResult method), 86</pre>	<pre>validate() (openff.evaluator.protocols.coordinates.SolvateExistingStructu method), 341</pre>
validate() (openff.evaluator.datasets.PhysicalProperty method), 95	<pre>validate() (openff.evaluator.protocols.forcefield.BaseBuildSystem     method), 352</pre>
walidate() (openff.evaluator.datasets.PhysicalPropertyD	avastidate() (openff.evaluator.protocols.forcefield.BuildLigParGenSystem
method) 140	method), 302
validate() (openff.evaluator.datasets.taproom.TaproomL	yalidate() (openff.evaluator.protocols.forcefield.BuildSmirnoffSystem
method) 150	method), 557
method) 146	Mail idate () (openff.evaluator.protocols.forcefield.BuildTLeapSystem         method), 368
<pre>validate() (openff.evaluator.layers.CalculationLayerRes</pre>	ult (openff.evaluator.protocols.gradients.ZeroGradients method), 373
method), 190	evalidate() (openff.evaluator.protocols.groups.ConditionalGroup
method) 191	meinoa), 579
validate() (openff.evaluator.layers.reweighting.Reweight	tingscheme () (openff.evaluator.protocols.groups.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionalGroup.ConditionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionActionAct
method), 202 validate() (openff evaluator layers simulation Simulation	ngelidate() (openff.evaluator.protocols.miscellaneous.AddValues
method) 100	meinoa), 584
<pre>validate() (openff.evaluator.layers.workflow.WorkflowCo method), 195</pre>	aliculate() (openff.evaluator.protocols.miscellaneous.DivideValue method), 399
validate() (openff.evaluator.properties.Density method), 103	<pre>validate() (openff.evaluator.protocols.miscellaneous.DummyProtocol</pre>
validate() (openff.evaluator.properties.DielectricConsta	nyalidate() (openff.evaluator.protocols.miscellaneous.FilterSubstanceByR
mathod) 110	method), 406
method) 114	gvalidate() (openff.evaluator.protocols.miscellaneous.MultiplyValue method), 394
validate() (openff.evaluator.properties.EnthalpyOfVapor method) 117	rizationate() (openff.evaluator.protocols.miscellaneous.SubtractValues method), 389
validate() (openff.evaluator.properties.ExcessMolarVolu	walidate() (openff.evaluator.protocols.miscellaneous.WeightByMoleFract
method) 107	methoa), 404
validate() (openff.evaluator.properties.HostGuestBindin	gAffinitiete() (openff.evaluator.protocols.openmm.OpenMMEnergyMinimise method), 420
method), 124	evalidate() (openff.evaluator.protocols.openmm.OpenMMEvaluateEnergi
method) 120	method), 452
validate() (openff.evaluator.protocols.analysis.AverageL	Dietectice () (openff.evaluator.protocols.openmm.OpenMMSimulation method), 426
validate() (openff.evaluator.protocols.analysis.AverageF	reeEnergies() (openff.evaluator.protocols.paprika.analysis.AnalyzeAPRPhas method), 474
method), 304 validate() (openff evaluator protocols analysis Average(	<i>byalidate</i> () (openff.evaluator.protocols.paprika.analysis.ComputeReference
method) 207	method), 484
validate() (openff.evaluator.protocols.analysis.BaseAver method) 201	age outset () (openff.evaluator.protocols.paprika.analysis.ComputeSymmetry method), 479
<pre>validate() (openff.evaluator.protocols.analysis.BaseDec</pre>	ovalidate(), (appenff.evaluator.protocols.paprika.coordinates.AddDummyAt method), 448
method), 318	DipoteMoments Malidate() (openff.evaluator.protocols.paprika.coordinates.PreparePullC
method), 313	method), 437

<pre>validate() (openff.evaluator.protocols.paprika.coordinate</pre>	s Prenare Reethned Caadinates
	validate() (openff.evaluator.storage.query.FreeEnergyDataQuery
validate() (openff.evaluator.protocols.paprika.restraints.	
	validate() (openff.evaluator.storage.query.SimulationDataQuery
<pre>validate() (openff.evaluator.protocols.paprika.restraints.</pre>	
	<pre>validate() (openff.evaluator.storage.query.SubstanceQuery</pre>
<pre>validate() (openff.evaluator.protocols.paprika.restraints.</pre>	
<i>method</i> ), 458	validate() (openff.evaluator.substances.Amount
validate() (openff.evaluator.protocols.paprika.restraints.	Generate <b>RelethsaR</b> estEdints
	validate() (openff.evaluator.substances.Component
<pre>validate() (openff.evaluator.protocols.reweighting.BaseE</pre>	
	validate() (openff.evaluator.substances.ExactAmount
validate() (openff.evaluator.protocols.reweighting.BaseM	
	validate() (openff.evaluator.substances.MoleFraction
validate() (openff.evaluator.protocols.reweighting.Conca	
	validate() (openff.evaluator.substances.Substance
<pre>validate() (openff.evaluator.protocols.reweighting.Conca</pre>	
	<pre>validate() (openff.evaluator.thermodynamics.ThermodynamicState</pre>
<pre>validate() (openff.evaluator.protocols.reweighting.Reweighting.</pre>	
<i>method</i> ), 516	validate() (openff.evaluator.workflow.Protocol
validate() (openff.evaluator.protocols.reweighting.Reweighting.	ghtObserv <b>ahdth</b> od), 265
method), 511	validate() (openff.evaluator.workflow.ProtocolGroup
<pre>validate() (openff.evaluator.protocols.simulation.BaseEn</pre>	
	validate() (openff.evaluator.workflow.schemas.ProtocolGroupSchema
validate() (openff.evaluator.protocols.simulation.BaseSin	
	validate() (openff.evaluator.workflow.schemas.ProtocolSchema
<pre>validate() (openff.evaluator.protocols.storage.UnpackSto</pre>	
	<pre>validate() (openff.evaluator.workflow.schemas.WorkflowSchema</pre>
<pre>validate() (openff.evaluator.protocols.yank.BaseYankPro</pre>	
<i>method</i> ), 538	<pre>validate() (openff.evaluator.workflow.WorkflowResult</pre>
<pre>validate() (openff.evaluator.protocols.yank.LigandRecept</pre>	torYankPr <b>ometloo</b> d), 260
<i>method</i> ), 549	value (openff.evaluator.datasets.PhysicalProperty
validate() (openff.evaluator.protocols.yank.SolvationYand	
	value (openff.evaluator.properties.Density attribute),
validate() (openff.evaluator.server.Batch method), 92	103
validate() (openff.evaluator.storage.data.BaseSimulation	
method), 228	attribute), 110
<pre>validate() (openff.evaluator.storage.data.BaseStoredData</pre>	
method), 220	tribute), 114
	value(openff.evaluator.properties.EnthalpyOfVaporization
<i>method</i> ), 223	attribute), 117
validate() (openff.evaluator.storage.data.HashableStored	Datue (openff.evaluator.properties.ExcessMolarVolume
<i>method</i> ), 222	attribute), 107
validate()(openff.evaluator.storage.data.ReplaceableDat	malue (openff.evaluator.properties.HostGuestBindingAffinity
method), 225	attribute), 124
validate() (openff.evaluator.storage.data.StoredFreeEner	
method), 234	attribute), 120
	<b>wElase</b> (openff.evaluator.protocols.analysis.AverageDielectricConstant
method), 231	attribute), 302
	$y_alue$ (openff.evaluator.protocols.analysis.AverageObservable
<i>method</i> ), 236	attribute), 297
<pre>validate() (openff.evaluator.storage.query.BaseSimulatio</pre>	nBluxQapey.ff.evaluator.protocols.analysis.BaseAverageObservable
<i>method</i> ), 241	attribute), 288
	walue (openff.evaluator.protocols.miscellaneous.DivideValue
	v · 1 / V/ 1

	attribute), 395	WeightB	yMoleFraction	(class	in
value(a	openff.evaluator.protocols.miscellaneous.MultiplyV attribute), 390	/alue	openff.evaluator.protoc 399	cols.miscellaneou	us),
value(a	openff.evaluator.protocols.miscellaneous.WeightBy attribute), 400	Malieghae	donvalue(openff.evalua attribute), 401	tor.protocols.mis	cellaneous.WeightByMol
value(a	openff.evaluator.protocols.reweighting.BaseMBAR attribute), 501	Pwimdolu_	index (openff.evaluator attribute), 465	r.protocols.paprii	ka.restraints.ApplyRestrai
value(a	openff.evaluator.protocols.reweighting.ReweightDia attribute), 517	ekeirteki <u>c</u> Ev		luator.utils.excep	tions.EvaluatorException
value(a	openff.evaluator.protocols.reweighting.ReweightOb attribute), 511	oswietat <u>bl</u> eer		luator.workflow.V	VorkflowException
value	( <i>openff.evaluator.substances.Amount attribute</i> ), 131		w (class in openff.evalue w_id(openff.evaluator.		
value	(openff.evaluator.substances.ExactAmount at- tribute), 133		attribute), 259 w_protocol()	(in	module
value	(openff.evaluator.substances.MoleFraction at- tribute), 135		openff.evaluator.workfl	ow), 272	ighting.ReweightingSchen
value(a	openff.evaluator.utils.observables.ObservableArray property), 568	v	attribute), 202		lation.SimulationSchema
value	(openff.evaluator.workflow.WorkflowResult at- tribute), 260		attribute), 199		flow.WorkflowCalculation
value_a	a (openff.evaluator.protocols.miscellaneous.Subtrat attribute), 386	ctValues	<i>attribute</i> ), 195 w_to_layer_result(		low.workflowCalculation
value_l	b (openff.evaluator.protocols.miscellaneous.Subtrac attribute), 386		(openff.evaluator.layer. static method), 200		weightingLayer
values	(openff.evaluator.protocols.analysis.AverageFreeE attribute), 304	nanggiksflo			ulation I aver
values	(openff.evaluator.protocols.miscellaneous.AddValu attribute), 381		static method), 197 w_to_layer_result(		munonLayer
values	() (openff.evaluator.utils.observables.ObservableF method), 570		(openff.evaluator.layer static method), 193		lowCalculationLayer
verbos	e (openff.evaluator.protocols.yank.BaseYankProtoc attribute), 537	<i>o</i> Workflo		(class workflow) 193	in
verbos	e (openff.evaluator.protocols.yank.LigandReceptor) attribute), 549	YMAARDIO		(class	in
verbos	e (openff.evaluator.protocols.yank.SolvationYankPr attribute), 557			<b>v</b>	orkflow)
verbos	<pre>e_packmol (openff.evaluator.protocols.coordinates attribute), 330</pre>	s.BuildCoc			
verbos	e_packmol (openff.evaluator.protocols.coordinates	s.SolvateEx	xistingStructure		•
volume	attribute), 341 s (openff.evaluator.protocols.analysis.AverageDiele	Workflo ectricCons		(class ow.schemas), 27	in 8
volume	attribute), 298 s (openff.evaluator.protocols.reweighting.Reweight				
W	attribute), 512	ZeroGra	openff.evaluator.protoc		
	ock_time_limit	zfill()	(openff.evaluator.st method), 253	orage.attributes.	FilePath
	(openff.evaluator.backends.QueueWorkerResource property), 206	ces			
water_n	model (openff.evaluator.protocols.forcefield.BuildL attribute), 362	igParGen.	System		
water_n	<pre>model (openff.evaluator.protocols.forcefield.BuildT attribute), 368</pre>	LeapSyste	m		