cognite-model-hosting Documentation

Release 0.1.2

Erlend Vollset

Apr 25, 2019

Contents

CHAPTER 1

Installation

To install this package run the following command

```
pip install cognite-model-hosting
```

CHAPTER 2

Overview

cognite-model-hosting is an open-source library containing utilities for working with data from the Cognite Data Platform (CDP) in the Model Hosting environment. Working with the data is split into two parts;

1) Specifying data, such as time series and files, using Data Specs
2) Fetching the described data using the Data Fetcher.

Data Specs are a collection of classes used to specify data in CDP. These specs are organized in a hierarchical way, so that they can be collected in a single object, sent easily around and passed to an instance of a DataFetcher. The Data Fetcher is then used to retrieve the specified data from the platform.

CHAPTER 3

Examples

```
from cognite.model_hosting.data_fetcher import DataFetcher
from cognite.model_hosting.data_spec import *

id1 = ...
id2 = ...
ts1 = TimeSeriesSpec(id=id1, start="3d-ago", end="now", granularity="1d", aggregate=
↪"average")
ts2 = TimeSeriesSpec(id=id2, start="3d-ago", end="now", granularity="1d", aggregate=
↪"max")
ds = DataSpec({"myts1": ts1, "myts2": ts2})
print(ds)

data_fetcher = DataFetcher(ds)
df = data_fetcher.time_series.fetch_dataframe(["myts1", "myts2"])
print(df)
```

3.1 Data Fetching

3.1.1 Data Fetcher

class cognite.model_hosting.data_fetcher.**DataFetcher**(*data_spec: Union[cognite.model_hosting.data_spec.data_spec.Dat Dict, str], api_key: str = None, project: str = None, base_url: str = None*)

Creates an instance of DataFetcher.

Parameters

- **data_spec** (DataSpec) – The data spec which describes the desired data.

- **api_key** (*str, optional*) – API key for authenticating against CDP. Defaults to the value of the environment variable "COGNITE_API_KEY".
- **project** (*str, optional*) – Project. Defaults to project of given API key.
- **base_url** (*str, optional*) – Base url to send requests to. Defaults to "https://api.cognitedata.com".

get_data_spec()
Returns a copy of the DataSpec passed to the DataFetcher.

Returns A copy of the DataSpec passed to the DataFetcher.

Return type *DataSpec*

files
Returns an instance of FileFetcher.

Returns An instance of FileFetcher.

Return type *FileFetcher*

time_series
Returns an instance of TimeSeriesFetcher.

Returns An instance of TimeSeriesFetcher.

Return type *TimeSeriesFetcher*

3.1.2 Time Series Fetcher

class cognite.model_hosting.data_fetcher.data_fetcher.**TimeSeriesFetcher**(*time_series_specs: Dict[str, cognite.model_hosting.data_sp*
cdp_client: cognite.model_hosting.data_fe

An object used for fetching time series data from CDP.

Attention: This class should never be instantiated directly, but rather accessed through the DataFetcher class.

Examples

Using the TimeSeriesFetcher:

```
from cognite.model_hosting.data_fetcher import DataFetcher

data_fetcher = DataFetcher(data_spec=...)

my_datapoints = data_fetcher.time_series.fetch_datapoints(alias="my_ts_1")
```

aliases
Returns the time series aliases defined in the data spec passed to the data fetcher.

Returns The time series aliases defined in the data spec passed to the data fetcher.

Return type List[str]

get_spec (*alias: str*) → cognite.model_hosting.data_spec.data_spec.TimeSeriesSpec
Returns the TimeSeriesSpec given by the alias.

Parameters **alias** (*str*) – The alias of the time series.

Returns The time series spec given by the alias.

Return type *TimeSeriesSpec*

fetch_dataframe (*aliases: List[str]*) → pandas.core.frame.DataFrame
Fetches a time-aligned dataframe of the time series specified by the provided aliases.

This method requires that all specified aliases must refer to time series aggregates with the same granularity, start, and end.

Parameters **aliases** (*List[str]*) – The list of aliases to retrieve a dataframe for.

Returns A pandas dataframe with the requested data.

Return type pandas.DataFrame

fetch_datapoints (*alias: Union[str, List[str]]*) → Union[pandas.core.frame.DataFrame, Dict[str, pandas.core.frame.DataFrame]]
Fetches dataframes for the time series specified by the aliases.

If a single alias is passed, a pandas DataFrame will be returned. If a list of aliases is passed, a dictionary which maps aliases to DataFrames is returned.

Parameters **alias** (*Union[List[str], str]*) – The alias(es) to retrieve data for.

Returns The requested dataframe(s).

Return type Union[pd.DataFrame, Dict[str, pd.DataFrame]

3.1.3 File Fetcher

class cognite.model_hosting.data_fetcher.data_fetcher.**FileFetcher** (*file_specs: Dict[str, cognite.model_hosting.data_spec.data_s cdp_client: cognite.model_hosting.data_fetcher._cli*

An object used for fetching files from CDP.

> **Attention:** This class should never be instantiated directly, but rather accessed through the DataFetcher class.

Examples

Using the FileFetcher:

```
from cognite.model_hosting.data_fetcher import DataFetcher

data_fetcher = DataFetcher(data_spec=...)

my_file = data_fetcher.files.fetch_to_memory(alias="my_file")
```

aliases
Returns the file aliases defined in the data spec passed to the data fetcher.

Returns The file aliases defined in the data spec passed to the data fetcher.

Return type List[str]

get_spec (*alias: str*) → cognite.model_hosting.data_spec.data_spec.FileSpec
Returns the FileSpec given by the alias

Parameters **alias** (*str*) – The alias of the file.

Returns The file spec given by the alias.

Return type *FileSpec*

fetch (*alias: Union[str, List[str]], directory: str = None*) → None
Fetches the file(s) given by the provided alias(es) to a given directory.

If provided, the directory must exist. If not provided, it will default to the current working directory.

If a single alias is passed, a pandas DataFrame will be returned. If a list of aliases is passed, a dictionary which maps aliases to DataFrames is returned.

Parameters

- **alias** (*Union[List[str], str]*) – The alias(es) to download file(s) for.
- **directory** (*str, optional*) – The directory to download the file(s) to.

Returns None

fetch_to_memory (*alias: Union[str, List[str]]*) → Union[bytes, Dict[str, bytes]]
Fetches the file(s) given by the provided alias(es) to memory.

If a list of aliases is passed, this method will return a dictionary mapping aliases to their respective file bytes.

Parameters **alias** (*Union[List[str], str]*) – The alias(es) to download file(s) for.

Returns The file(s).

Return type Union[bytes, Dict[str, bytes]]

3.1.4 Exceptions

exception cognite.model_hosting.data_fetcher.exceptions.**ApiKeyError**
Raised if the provided API key is missing or invalid.

exception cognite.model_hosting.data_fetcher.exceptions.**DataFetcherHttpError** (*message*, *code=None*, *x_request_id=None*, *extra=None*)
Raised if an HTTP Error occurred while processing your request.

Parameters

- **message** (*str*) – The error message produced by the API.
- **code** (*int*) – The error code produced by the failure.
- **x_request_id** (*str*) – The request-id generated for the failed request.
- **extra** (*Dict*) – A dict of any additional information.

exception cognite.model_hosting.data_fetcher.exceptions.**DirectoryDoesNotExist**(*directory*)
Raised if the specified directory does not exist.

exception cognite.model_hosting.data_fetcher.exceptions.**InvalidAlias**(*alias*)
Raised if an invalid alias is specified.

exception cognite.model_hosting.data_fetcher.exceptions.**InvalidFetchRequest**
Raised if an invalid fetch request is issued.

For example if a request is issued for a time-aligned dataframe where the specified starts/ends or granularities of the time series are not the same.

3.2 Data Specs

3.2.1 Data Spec

class cognite.model_hosting.data_spec.**DataSpec**(*time_series: Dict[str, cognite.model_hosting.data_spec.data_spec.TimeSeriesSpec] = None*, *files: Dict[str, cognite.model_hosting.data_spec.data_spec.FileSpec] = None*)
Creates a DataSpec.

This object collects all data specs specific for a given resource type into a single object which can be passed to the DataFetcher. It includes aliases for all specs so that they may be referenced by a user-defined shorthand and abstracted away from specific resources.

Parameters

- **time_series** (*Dict[str, TimeSeriesSpec]*) – A dictionary mapping aliases to TimeSeriesSpecs.
- **files** (*Dict[str, FileSpec]*) – A dicionary mapping aliases to FileSpecs.

copy()
Returns a copy of the data spec.

Raises SpecValidationError – If the spec is not valid.

dump()
Dumps the data spec into a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec as a Python data structure.

Return type Dict

classmethod from_json(*s: str*)
Loads the data spec from a json representation.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec object.

classmethod load(*data*)
Loads the data from a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec object.

to_json()
Returns a json representation of the data spec.

Raises SpecValidationError – If the spec is not valid.

Returns The json representation of the data spec.

Return type str

validate()
Checks whether or not the data spec is valid.

Raises SpecValidationError – If the spec is not valid.

3.2.2 Time Series Spec

class cognite.model_hosting.data_spec.**TimeSeriesSpec**(*id: int, start: Union[int, str, datetime.datetime], end: Union[int, str, datetime.datetime], aggregate: str = None, granularity: str = None, include_outside_points: bool = None*)
Creates a time series spec.

If the granularity and aggregate parameters are omitted, the TimeSeriesSpec specifies raw data.

Parameters

- **id** (*int*) – The id of the time series.
- **start** (*Union[str, int, datetime]*) – The (inclusive) start of the time series. Can be either milliseconds since epoch,
- **format** (*time-ago*) –
- **end** (*Union[str, int, datetime]*) – The (exclusive) end of the time series. Same format as start. Can also be set to "now".
- **aggregate** (*str, optional*) – The aggregate function to apply to the time series.
- **granularity** (*str, optional*) – Granularity of the datapoints. e.g. "1m", "2h", or "3d".
- **include_outside_points** (*bool*) – Whether or not to include the first point before and after start and end. Can only be used with raw data.

copy()
Returns a copy of the data spec.

Raises SpecValidationError – If the spec is not valid.

dump()
Dumps the data spec into a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec as a Python data structure.

Return type Dict

classmethod from_json(*s: str*)
Loads the data spec from a json representation.

Raises `SpecValidationError` – If the spec is not valid.

Returns The data spec object.

classmethod load(*data*)
Loads the data from a Python data structure.

Raises `SpecValidationError` – If the spec is not valid.

Returns The data spec object.

to_json()
Returns a json representation of the data spec.

Raises `SpecValidationError` – If the spec is not valid.

Returns The json representation of the data spec.

Return type str

validate()
Checks whether or not the data spec is valid.

Raises `SpecValidationError` – If the spec is not valid.

3.2.3 File Spec

class cognite.model_hosting.data_spec.**FileSpec**(*id: int*)
Creates a file spec.

Parameters **id**(*int*) – The id of the file.

copy()
Returns a copy of the data spec.

Raises `SpecValidationError` – If the spec is not valid.

dump()
Dumps the data spec into a Python data structure.

Raises `SpecValidationError` – If the spec is not valid.

Returns The data spec as a Python data structure.

Return type Dict

classmethod from_json(*s: str*)
Loads the data spec from a json representation.

Raises `SpecValidationError` – If the spec is not valid.

Returns The data spec object.

classmethod load(*data*)
Loads the data from a Python data structure.

Raises `SpecValidationError` – If the spec is not valid.

Returns The data spec object.

to_json()
Returns a json representation of the data spec.

Raises `SpecValidationError` – If the spec is not valid.

Returns The json representation of the data spec.

Return type str

validate()
Checks whether or not the data spec is valid.

Raises SpecValidationError – If the spec is not valid.

3.2.4 Exceptions

exception cognite.model_hosting.data_spec.exceptions.**SpecValidationError**(*errors*)
Raised if a data spec is invalid.

Parameters **errors** (*Dict*) – A dictionary describing which fields are invalid and why.

3.3 Schedules

3.3.1 Data Specs

Schedule Data Spec

class cognite.model_hosting.data_spec.**ScheduleDataSpec**(*input: cognite.model_hosting.data_spec.data_spec.ScheduleInp output: cognite.model_hosting.data_spec.data_spec.ScheduleOu stride: Union[int, str, datetime.timedelta], window_size: Union[int, str, datetime.timedelta], start: Union[int, str, datetime.datetime] = 'now', slack: Union[int, str, datetime.timedelta] = 0*)
Creates a ScheduleDataSpec.

This spec defines the input and output data for a given schedule, as well as how the hosting environment should feed the specified data to your model. This is done by specifying window size, a stride, and start time for the schedule.

Parameters

- **input** (ScheduleInputSpec) – A schedule input spec describing input for a model.
- **output** (ScheduleOutputSpec) – A schedule output spec describing output for a model.
- **stride** (*Union[int, str, timedelta]*) – The interval at which predictions will be made. Can be either milliseconds, a timedelta object, or a time-string (e.g. "1h", "10d", "120s").
- **window_size** (*Union[int, str, timedelta]*) – The size of each prediction window, i.e. how long back in time a prediction will look. Same format as stride.
- **start** (*Union[int, str, datetime]*) – When the first prediction will be made.
- **slack** (*Union[int, str, timedelta]*) – How long back in time input changes will trigger new predictions

get_instances(*start: Union[int, str, datetime.datetime], end: Union[int, str, datetime.datetime]*) → List[cognite.model_hosting.data_spec.data_spec.DataSpec]

Returns the DataSpec objects describing the prediction windows executed between start and end.

Parameters

- **start** (*Union[str, int, datetime]*) – The start of the time period. Can be either milliseconds since epoch, time-ago format (e.g. "1d-ago"), or a datetime object.
- **end** (*Union[str, int, datetime]*) – The end of the time period. Same format as start. Can also be set to "now".

Returns List of DataSpec objects, one for each prediction window.

Return type List[*DataSpec*]

get_execution_timestamps(*start: Union[int, str, datetime.datetime], end: Union[int, str, datetime.datetime]*) → List[int]

Returns a list of timestamps indicating when each prediction will be executed.

This corresponds to the end of each DataSpec returned from get_instances().

Parameters

- **start** (*Union[str, int, datetime]*) – The start of the time period. Can be either milliseconds since epoch, time-ago format (e.g. "1d-ago"), or a datetime object.
- **end** (*Union[str, int, datetime]*) – The end of the time period. Same format as start. Can also be set to "now".

Returns A list of timestamps.

Return type List[int]

copy()

Returns a copy of the data spec.

Raises SpecValidationError – If the spec is not valid.

dump()

Dumps the data spec into a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec as a Python data structure.

Return type Dict

classmethod from_json(*s: str*)

Loads the data spec from a json representation.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec object.

classmethod load(*data*)

Loads the data from a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec object.

to_json()

Returns a json representation of the data spec.

Raises SpecValidationError – If the spec is not valid.

Returns The json representation of the data spec.

Return type str

validate()
Checks whether or not the data spec is valid.

Raises SpecValidationError – If the spec is not valid.

Schedule Input Spec

class cognite.model_hosting.data_spec.**ScheduleInputSpec**(*time_series: Dict[str, cognite.model_hosting.data_spec.data_spec.ScheduleI*
= None)
Creates a ScheduleInputSpec.

The provided aliases must be the same as the input fields defined on the model.

Parameters **time_series** (*Dict[str, ScheduleInputTimeSeriesSpec]*) – A dictionary mapping aliases to ScheduleInputTimeSeriesSpec objects.

copy()
Returns a copy of the data spec.

Raises SpecValidationError – If the spec is not valid.

dump()
Dumps the data spec into a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec as a Python data structure.

Return type Dict

classmethod from_json(*s: str*)
Loads the data spec from a json representation.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec object.

classmethod load(*data*)
Loads the data from a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec object.

to_json()
Returns a json representation of the data spec.

Raises SpecValidationError – If the spec is not valid.

Returns The json representation of the data spec.

Return type str

validate()
Checks whether or not the data spec is valid.

Raises SpecValidationError – If the spec is not valid.

Schedule Input Time Series

class cognite.model_hosting.data_spec.**ScheduleInputTimeSeriesSpec**(*id: int*, *aggregate: str = None*, *granularity: str = None*, *include_outside_points: bool = None*)

Creates a ScheduleOutputTimeSeriesSpec.

This object defines the time series a schedule should read from.

If the granularity and aggregate parameters are omitted, the spec specifies raw data.

Parameters

- **id** (*int*) – The id of the output time series.
- **aggregate** (*str, optional*) – The aggregate function to apply to the time series.
- **granularity** (*str, optional*) – Granularity of the datapoints. e.g. "1m", "2h", or "3d".
- **include_outside_points** (*bool, optional*) – Whether or not to include the first point before and after start and end. Can only be used with raw data.

copy()

Returns a copy of the data spec.

Raises SpecValidationError – If the spec is not valid.

dump()

Dumps the data spec into a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec as a Python data structure.

Return type Dict

classmethod from_json(*s: str*)

Loads the data spec from a json representation.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec object.

classmethod load(*data*)

Loads the data from a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec object.

to_json()

Returns a json representation of the data spec.

Raises SpecValidationError – If the spec is not valid.

Returns The json representation of the data spec.

Return type str

validate()
Checks whether or not the data spec is valid.

Raises SpecValidationError – If the spec is not valid.

Schedule Output Spec

class cognite.model_hosting.data_spec.**ScheduleOutputSpec**(*time_series: Dict[str, cognite.model_hosting.data_spec.data_spec.ScheduleOutputTimeSeriesSpec] = None*)

Creates a ScheduleOutputSpec.

The provided aliases must be the same as the output fields defined on the model.

Parameters **time_series** (*Dict[str, ScheduleInputTimeSeriesSpec]*) – A dictionary mapping aliases to ScheduleOutputTimeSeriesSpec objects.

copy()
Returns a copy of the data spec.

Raises SpecValidationError – If the spec is not valid.

dump()
Dumps the data spec into a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec as a Python data structure.

Return type Dict

classmethod from_json(*s: str*)
Loads the data spec from a json representation.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec object.

classmethod load(*data*)
Loads the data from a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec object.

to_json()
Returns a json representation of the data spec.

Raises SpecValidationError – If the spec is not valid.

Returns The json representation of the data spec.

Return type str

validate()
Checks whether or not the data spec is valid.

Raises SpecValidationError – If the spec is not valid.

Schedule Output Time Series

class cognite.model_hosting.data_spec.**ScheduleOutputTimeSeriesSpec**(*id: int*, *offset: Union[int, str, datetime.timedelta] = 0*)

Creates a ScheduleOutputTimeSeriesSpec.

This object defines the time series a schedule should write to. You need to specify an offset which defines where in time your schedule can write data to for a given window. Offset defaults to 0, meaning that your schedule can write to the same time window which it was feeded data from.

Parameters

- **id** (*int*) – The id of the output time series.
- **offset** (*Union[int, str, timedelta], optional*) – The offset of the window to which your schedule is allowed to write data.

copy()

Returns a copy of the data spec.

Raises SpecValidationError – If the spec is not valid.

dump()

Dumps the data spec into a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec as a Python data structure.

Return type Dict

classmethod from_json(*s: str*)

Loads the data spec from a json representation.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec object.

classmethod load(*data*)

Loads the data from a Python data structure.

Raises SpecValidationError – If the spec is not valid.

Returns The data spec object.

to_json()

Returns a json representation of the data spec.

Raises SpecValidationError – If the spec is not valid.

Returns The json representation of the data spec.

Return type str

validate()

Checks whether or not the data spec is valid.

Raises SpecValidationError – If the spec is not valid.

3.3.2 Helpers

class cognite.model_hosting.schedules.schedules.**ScheduleOutput**(*output: Dict*)
Helper class for parsing and converting output from scheduled predictions.

Parameters **output** (*Dict*) – The output returned from the scheduled prediction.

get_dataframe(*alias: Union[str, List[str]]*) → pandas.core.frame.DataFrame
Returns a time-aligned dataframe of the specified alias(es).

Assumes that all aliases specify output time series with matching timestamps.

Parameters **alias** (*Union[str, List[str]]*) – alias or list of aliases

Returns The dataframe containing the time series for the specified alias(es).

Return type pd.DataFrame

get_datapoints(*alias: Union[str, List[str]]*) → Union[pandas.core.frame.DataFrame, Dict[str, pandas.core.frame.DataFrame]]
Returns the dataframes for the specified alias(es).

Parameters **alias** (*Union[str, List[str]]*) – alias or list of aliases.

Returns

A single dataframe if a single alias has been specified. Or a dictionary mapping alias to dataframe if a list of aliases has been provided.

Return type Union[pd.DataFrame, Dict[str, pd.DataFrame]

cognite.model_hosting.schedules.schedules.**to_output**(*dataframe: Union[pandas.core.frame.DataFrame, List[pandas.core.frame.DataFrame]]*)
Converts your data to a json serializable output format complying with the schedules feature.

Parameters **(Union[List[pd.DataFrame, pd.DataFrame]]** (*dataframe*) – A dataframe or list of dataframes.

Returns The data on a json serializable and schedules compliant output format.

Return type Dict

Examples

The correct output format looks like this:

```
{
    "timeSeries":
        {
            "my-alias-1": [(t0, p0), (t1, p1), ...],
            "my-alias-2": [(t0, p0), (t1, p1), ...],
        }
}
```

3.3.3 Exceptions

exception cognite.model_hosting.schedules.exceptions.**DataframeMissingTimestampColumn**
Raised if trying to convert a dataframe without a timestamp column to scheduled output format.

exception cognite.model_hosting.schedules.exceptions.**DuplicateAliasInScheduledOutput**
Raised when an alias is passed more than once when converting to scheduled output format.

exception cognite.model_hosting.schedules.exceptions.**InvalidScheduleOutputFormat**(*errors*)
Raised if the scheduled output is on an invalid format.

Python Module Index

C

Index