cognite-sdk-python Documentation

Release 1.6.0

Cognite

May 19, 2020

Contents

This is the Cognite Python SDK for developers and data scientists working with Cognite Data Fusion (CDF). The package is tightly integrated with pandas, and helps you work easily and efficiently with data in Cognite Data Fusion (CDF).

CHAPTER 1

Installation

To install this package:

```
pip install cognite-sdk
```

To install this package without the pandas and NumPy support:

```
pip install cognite-sdk-core
```

CHAPTER 2

Contents

2.1 Quickstart

2.1.1 Authenticate

The preferred way to authenticate against the Cognite API is by setting the `COGNITE_API_KEY` environment variable. All examples in this documentation require that the variable has been set.

```
$ export COGNITE_API_KEY = <your-api-key>
```

You can also pass your API key directly to the CogniteClient.

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient(api_key="<your-api-key>", client_name="<your-client-name>")
```

2.1.2 Instantiate a new client

Use this code to instantiate a client and get your login status. CDF returns an object with attributes that describe which project and service account your API key belongs to. The `client_name` is an user-defined string intended to give the client a unique identifier. You can provide the `client_name` through the `COGNITE_CLIENT_NAME` environment variable or by passing it directly to the `CogniteClient` constructor. All examples in this documentation assume that `COGNITE_CLIENT_NAME` has been set.

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> status = c.login.status()
```

Read more about the *CogniteClient* and the functionality it exposes below.

2.1.3 Discover time series

For the next examples, you will need to supply ids for the time series that you want to retrieve. You can find some ids by listing the available time series. Limits for listing resources default to 25, so the following code will return the first 25 time series resources.

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> ts_list = c.time_series.list(include_metadata=False)
```

2.1.4 Plot time series

There are several ways of plotting a time series you have fetched from the API. The easiest is to call `.plot()` on the returned `TimeSeries` or `TimeSeriesList` objects. By default, this plots the raw data points for the last 24 hours. If there are no data points for the last 24 hours, `plot` will throw an exception.

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> my_time_series = c.time_series.retrieve(id=<time-series-id>)
>>> my_time_series.plot()
```

You can also pass arguments to the `.plot()` method to change the start, end, aggregates, and granularity of the request.

```
>>> my_time_series.plot(start="365d-ago", end="now", aggregates=["average"],␣
↪granularity="1d")
```

The `Datapoints` and `DatapointsList` objects that are returned when you fetch data points, also have `.plot()` methods you can use to plot the data.

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> my_datapoints = c.datapoints.retrieve(
...                     id=[<time-series-ids>],
...                     start="10d-ago",
...                     end="now",
...                     aggregates=["max"],
...                     granularity="1h"
...                 )
>>> my_datapoints.plot()
```

Note: To use the `.plot()` functionality you need to install `matplotlib`.

2.1.5 Create an asset hierarchy

CDF organizes digital information about the physical world. Assets are digital representations of physical objects or groups of objects, and assets are organized into an asset hierarchy. For example, an asset can represent a water pump which is part of a subsystem on an oil platform.

At the top of an asset hierarchy is a root asset (e.g., the oil platform). Each project can have multiple root assets. All assets have a name and a parent asset. No assets with the same parent can have the same name.

To create a root asset (an asset without a parent), omit the parent ID when you post the asset to the API. To make an asset a child of an existing asset, you must specify a parent ID.

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import Asset
>>> c = CogniteClient()
>>> my_asset = Asset(name="my first asset", parent_id=123)
>>> c.assets.create(my_asset)
```

To post an entire asset hierarchy, you can describe the relations within your asset hierarchy using the `external_id` and `parent_external_id` attributes on the `Asset` object. You can post an arbitrary number of assets, and the SDK will split the request into multiple requests. To make sure that the assets are posted in the correct order, you can use the .create_hierarchy() function, which takes care of the sorting before splitting the request into smaller chunks. However, note that the .create_hierarchy() function requires the external_id property to be set for all assets.

This example shows how to post a three levels deep asset hierarchy consisting of three assets.

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import Asset
>>> c = CogniteClient()
>>> root = Asset(name="root", external_id="1")
>>> child = Asset(name="child", external_id="2", parent_external_id="1")
>>> descendant = Asset(name="descendant", external_id="3", parent_external_id="2")
>>> c.assets.create_hierarchy([root, child, descendant])
```

Wrap the .create_hierarchy() call in a try-except to get information if posting the assets fails:

- Which assets were posted. (The request yielded a 201.)
- Which assets may have been posted. (The request yielded 5xx.)
- Which assets were not posted. (The request yielded 4xx, or was a descendant of another asset which may or may not have been posted.)

```
>>> from cognite.client.exceptions import CogniteAPIError
>>> try:
...     c.create_hierarchy([root, child, descendant])
>>> except CogniteAPIError as e:
...     assets_posted = e.successful
...     assets_may_have_been_posted = e.unknown
...     assets_not_posted = e.failed
```

2.1.6 Retrieve all events related to an asset subtree

Assets are used to connect related data together, even if the data comes from different sources; Time series of data points, events and files are all connected to one or more assets. A pump asset can be connected to a time series measuring pressure within the pump, as well as events recording maintenance operations, and a file with a 3D diagram of the pump.

To retrieve all events related to a given subtree of assets, we first fetch the subtree under a given asset using the `.subtree()` method. This returns an `AssetList` object, which has a `.events()` method. This method will return events related to any asset in the `AssetList`.

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import Asset
>>> c = CogniteClient()
>>> subtree_root_asset="some-external-id"
```

(continues on next page)

(continued from previous page)

```
>>> subtree = c.assets.retrieve(external_id=subtree_root_asset).subtree()
>>> related_events = subtree.events()
```

You can use the same pattern to retrieve all time series or files related to a set of assets.

```
>>> related_files = subtree.files()
>>> related_time_series = subtree.time_series()
```

2.2 Settings

2.2.1 Client configuration

You can pass configuration arguments directly to the `CogniteClient` constructor, for example to configure the base url of your requests and additional headers. For a list of all configuration arguments, see the *CogniteClient* class definition.

2.2.2 Environment configuration

You can set default configurations with these environment variables:

```
# Can be overrided by Client Configuration
$ export COGNITE_API_KEY = <your-api-key>
$ export COGNITE_PROJECT = <your-default-project>
$ export COGNITE_BASE_URL = http://<host>:<port>
$ export COGNITE_CLIENT_NAME = <user-defined-client-or-app-name>
$ export COGNITE_MAX_WORKERS = <number-of-workers>
$ export COGNITE_TIMEOUT = <num-of-seconds>

# Global Configuration
$ export COGNITE_DISABLE_PYPI_VERSION_CHECK = "1"
$ export COGNITE_DISABLE_GZIP = "1"
$ export COGNITE_MAX_RETRIES = <number-of-retries>
$ export COGNITE_MAX_RETRY_BACKOFF = <number-of-seconds>
$ export COGNITE_MAX_CONNECTION_POOL_SIZE = <number-of-connections-in-pool>
$ export COGNITE_STATUS_FORCELIST = "429,502,503"
```

2.2.3 Concurrency and connection pooling

This library does not expose API limits to the user. If your request exceeds API limits, the SDK splits your request into chunks and performs the sub-requests in parallel. To control how many concurrent requests you send to the API, you can either pass the `max_workers` attribute when you instantiate the `CogniteClient` or set the `COGNITE_MAX_WORKERS` environment variable.

If you are working with multiple instances of `CogniteClient`, all instances will share the same connection pool. If you have several instances, you can increase the max connection pool size to reuse connections if you are performing a large amount of concurrent requests. You can increase the max connection pool size by setting the `COGNITE_MAX_CONNECTION_POOL_SIZE` environment variable.

2.3 Extensions and core library

2.3.1 Pandas integration

The SDK is tightly integrated with the pandas library. You can use the `.to_pandas()` method on pretty much any object and get a pandas data frame describing the data.

This is particularly useful when you are working with time series data and with tabular data from the Raw API.

2.3.2 Matplotlib integration

You can use the `.plot()` method on any time series or data points result that the SDK returns. The method takes keyword arguments which are passed on to the underlying matplotlib plot function, allowing you to configure for example the size and layout of your plots.

You need to install the matplotlib package manually:

```
$ pip install matplotlib
```

2.3.3 `cognite-sdk` vs. `cognite-sdk-core`

If your application doesn't require the functionality from the `pandas` or `numpy` dependencies, you should install the `cognite-sdk-core` library.

The two libraries are exactly the same, except that `cognite-sdk-core` does not specify `pandas` or `numpy` as dependencies. This means that `cognite-sdk-core` only has a subset of the features available through the `cognite-sdk` package. If you attempt to use functionality that `cognite-sdk-core` does not support, a `CogniteImportError` is raised.

2.4 API

2.4.1 CogniteClient

class cognite.client.**CogniteClient**(*api_key: str = None, project: str = None, client_name: str = None, base_url: str = None, max_workers: int = None, headers: Dict[str, str] = None, timeout: int = None, token: Union[str, Callable[[], str], None] = None, disable_pypi_version_check: Optional[bool] = None, debug: bool = False*)

Main entrypoint into Cognite Python SDK.

All services are made available through this object. See examples below.

Parameters

- **api_key** (*str*) – API key
- **project** (*str*) – Project. Defaults to project of given API key.
- **client_name** (*str*) – A user-defined name for the client. Used to identify number of unique applications/scripts running on top of CDF.
- **base_url** (*str*) – Base url to send requests to. Defaults to "https://api.cognitedata.com"

- **max_workers** (*int*) – Max number of workers to spawn when parallelizing data fetching. Defaults to 10.
- **headers** (*Dict*) – Additional headers to add to all requests.
- **timeout** (*int*) – Timeout on requests sent to the api. Defaults to 30 seconds.
- **token** (*Union[str, Callable[[], str]]*) – A jwt or method which takes no arguments and returns a jwt to use for authentication. This will override any api-key set.
- **disable_pypi_version_check** (*bool*) – Don't check for newer versions of the SDK on client creation
- **debug** (*bool*) – Configures logger to log extra request details to stderr.

get (*url: str*, *params: Dict[str, Any] = None*, *headers: Dict[str, Any] = None*)
Perform a GET request to an arbitrary path in the API.

post (*url: str, json: Dict[str, Any], params: Dict[str, Any] = None, headers: Dict[str, Any] = None*)
Perform a POST request to an arbitrary path in the API.

put (*url: str*, *json: Dict[str, Any] = None*, *headers: Dict[str, Any] = None*)
Perform a PUT request to an arbitrary path in the API.

delete (*url: str*, *params: Dict[str, Any] = None*, *headers: Dict[str, Any] = None*)
Perform a DELETE request to an arbitrary path in the API.

version
Returns the current SDK version.

Returns The current SDK version

Return type str

config
Returns a config object containing the configuration for the current client.

Returns The configuration object.

Return type ClientConfig

2.4.2 Authentication

Get login status

LoginAPI.**status**() → cognite.client.data_classes.login.LoginStatus
Check login status

Returns The login status of the current api key.

Return type *LoginStatus*

Examples

Check the current login status and get the project:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> login_status = c.login.status()
>>> project = login_status.project
```

Data classes

class cognite.client.data_classes.login.**LoginStatus**(*user: str*, *project: str*, *logged_in: bool*, *project_id: int*, *api_key_id: int*)

Bases: *cognite.client.data_classes._base.CogniteResponse*

Current login status

Parameters

- **user** (*str*) – Current user.
- **logged_in** (*bool*) – Is user logged in.
- **project** (*str*) – Current project.
- **project_id** (*int*) – Current project id.
- **api_key_id** (*int*) – Current api key id.

dump(*camel_case: bool = False*) → Dict[str, Any]

Dump the instance into a json serializable python data type.

Parameters **camel_case** (*bool*) – Use camelCase for attribute names. Defaults to False.

Returns A dictionary representation of the instance.

Return type Dict[str, Any]

to_pandas()

2.4.3 Assets

Retrieve an asset by id

AssetsAPI.**retrieve**(*id: Optional[int] = None*, *external_id: Optional[str] = None*) → Optional[cognite.client.data_classes.assets.Asset]

Retrieve a single asset by id.

Parameters

- **id** (*int, optional*) – ID
- **external_id** (*str, optional*) – External ID

Returns Requested asset or None if it does not exist.

Return type Optional[*Asset*]

Examples

Get asset by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.assets.retrieve(id=1)
```

Get asset by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.assets.retrieve(external_id="1")
```

Retrieve multiple assets by id

AssetsAPI.**retrieve_multiple**(*ids: Optional[List[int]] = None*, *external_ids: Optional[List[str]] = None*, *ignore_unknown_ids: bool = False*) → cognite.client.data_classes.assets.AssetList

Retrieve multiple assets by id.

Parameters

- **ids** (*List[int], optional*) – IDs
- **external_ids** (*List[str], optional*) – External IDs
- **ignore_unknown_ids** (*bool*) – Ignore IDs and external IDs that are not found rather than throw an exception.

Returns The requested assets.

Return type *AssetList*

Examples

Get assets by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.assets.retrieve_multiple(ids=[1, 2, 3])
```

Get assets by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.assets.retrieve_multiple(external_ids=["abc", "def"], ignore_unknown_
↪ids=True)
```

Retrieve an asset subtree

AssetsAPI.**retrieve_subtree**(*id: int = None*, *external_id: str = None*, *depth: int = None*) → cognite.client.data_classes.assets.AssetList

Retrieve the subtree for this asset up to a specified depth.

Parameters

- **id** (*int*) – Id of the root asset in the subtree.
- **external_id** (*str*) – External id of the root asset in the subtree.
- **depth** (*int*) – Retrieve assets up to this depth below the root asset in the subtree. Omit to get the entire subtree.

Returns The requested assets.

Return type *AssetList*

List assets

AssetsAPI.**list**(*name: str = None, parent_ids: List[int] = None, parent_external_ids: List[str] = None, root_ids: List[int] = None, root_external_ids: List[str] = None, asset_subtree_ids: List[int] = None, asset_subtree_external_ids: List[str] = None, data_set_ids: List[int] = None, data_set_external_ids: List[str] = None, metadata: Dict[str, str] = None, source: str = None, created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, last_updated_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, root: bool = None, external_id_prefix: str = None, aggregated_properties: List[str] = None, partitions: int = None, limit: int = 25*) → cognite.client.data_classes.assets.AssetList

List assets

Parameters

- **name** (*str*) – Name of asset. Often referred to as tag.
- **parent_ids** (*List[int]*) – Return only the direct descendants of the specified assets.
- **parent_external_ids** (*List[str]*) – Return only the direct descendants of the specified assets.
- **root_ids** (*List[int], optional*) – List of root ids ids to filter on.
- **root_external_ids** (*List[str], optional*) – List of root external ids to filter on.
- **asset_subtree_ids** (*List[int]*) – List of asset subtrees ids to filter on.
- **asset_subtree_external_ids** (*List[str]*) – List of asset subtrees external ids to filter on.
- **data_set_ids** (*List[int]*) – Return only assets in the specified data sets with these ids.
- **data_set_external_ids** (*List[str]*) – Return only assets in the specified data sets with these external ids.
- **metadata** (*Dict[str, str]*) – Custom, application specific metadata. String key -> String value.
- **source** (*str*) – The source of this asset.
- **created_time** (*Union[Dict[str, int], TimestampRange]*) – Range between two timestamps. Possible keys are *min* and *max*, with values given as time stamps in ms.
- **last_updated_time** (*Union[Dict[str, int], TimestampRange]*) – Range between two timestamps. Possible keys are *min* and *max*, with values given as time stamps in ms.
- **root** (*bool*) – filtered assets are root assets or not.
- **external_id_prefix** (*str*) – Filter by this (case-sensitive) prefix for the external ID.
- **aggregated_properties** (*List[str]*) – Set of aggregated properties to include.
- **partitions** (*int*) – Retrieve assets in parallel using this number of workers. Also requires *limit=None* to be passed.
- **limit** (*int, optional*) – Maximum number of assets to return. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns List of requested assets

Return type *AssetList*

Examples

List assets:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> asset_list = c.assets.list(limit=5)
```

Iterate over assets:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for asset in c.assets:
...     asset # do something with the asset
```

Iterate over chunks of assets to reduce memory load:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for asset_list in c.assets(chunk_size=2500):
...     asset_list # do something with the assets
```

Aggregate assets

AssetsAPI.**aggregate**(*filter: Union[cognite.client.data_classes.assets.AssetFilter, Dict[KT, VT]] = None*) → List[cognite.client.data_classes.assets.AssetAggregate]

Aggregate assets

Parameters **filter** (*Union[AssetFilter, Dict]*) – Filter on assets filter with exact match

Returns List of asset aggregates

Return type List[*AssetAggregate*]

Examples

Aggregate assets:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> aggregate_by_prefix = c.assets.aggregate(filter={"external_id_prefix": "prefix
↪"})
```

Search for assets

AssetsAPI.**search**(*name: str = None, description: str = None, query: str = None, filter: Union[cognite.client.data_classes.assets.AssetFilter, Dict[KT, VT]] = None, limit: int = 100*) → cognite.client.data_classes.assets.AssetList

Search for assets Primarily meant for human-centric use-cases and data exploration, not for programs, since matching and ordering may change over time. Use the *list* function if stable or exact matches are required.

Parameters

- **name** (*str*) – Fuzzy match on name.
- **description** (*str*) – Fuzzy match on description.
- **query** (*str*) – Whitespace-separated terms to search for in assets. Does a best-effort fuzzy search in relevant fields (currently name and description) for variations of any of the search terms, and orders results by relevance.
- **filter** (*Union[AssetFilter, Dict]*) – Filter to apply. Performs exact match on these fields.
- **limit** (*int*) – Maximum number of results to return.

Returns List of requested assets

Return type *AssetList*

Examples

Search for assets by fuzzy search on name:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.assets.search(name="some name")
```

Search for assets by exact search on name:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.assets.search(filter={"name": "some name"})
```

Search for assets by improved multi-field fuzzy search:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.assets.search(query="TAG 30 XV")
```

Search for assets using multiple filters, finding all assets with name similar to *xyz* with parent asset *123* or *456* with source *some source*:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.assets.search(name="xyz",filter={"parent_ids": [123,456],"source":
↪"some source"})
```

Create assets

AssetsAPI.**create**(*asset: Union[cognite.client.data_classes.assets.Asset, List[cognite.client.data_classes.assets.Asset]]*) → Union[cognite.client.data_classes.assets.Asset, cognite.client.data_classes.assets.AssetList]

Create one or more assets.

You can create an arbitrary number of assets, and the SDK will split the request into multiple requests. When specifying parent-child relation between assets using *parentExternalId* the link will be resvoled into an internal ID and stored as *parentId*.

Parameters **asset** (*Union[Asset, List[Asset]]*) – Asset or list of assets to create.

Returns Created asset(s)

Return type Union[*Asset*, *AssetList*]

Examples

Create new assets:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import Asset
>>> c = CogniteClient()
>>> assets = [Asset(name="asset1"), Asset(name="asset2")]
>>> res = c.assets.create(assets)
```

Create asset hierarchy

AssetsAPI.**create_hierarchy**(*assets: List[cognite.client.data_classes.assets.Asset]*) → cognite.client.data_classes.assets.AssetList

Create asset hierarchy. Like the create() method, when posting a large number of assets, the IDE will split the request into smaller requests. However, create_hierarchy() will additionally make sure that the assets are posted in correct order. The ordering is determined from the external_id and parent_external_id properties of the assets, and the external_id is therefore required for all assets. Before posting, it is checked that all assets have a unique external_id and that there are no circular dependencies.

Parameters **assets** (*List[Asset]*) – List of assets to create. Requires each asset to have a unique external id.

Returns Created asset hierarchy

Return type *AssetList*

Examples

Create asset hierarchy:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import Asset
>>> c = CogniteClient()
>>> assets = [Asset(external_id="root", name="root"), Asset(external_id="child1",␣
↪parent_external_id="root", name="child1"), Asset(external_id="child2", parent_
↪external_id="root", name="child2")]
>>> res = c.assets.create_hierarchy(assets)
```

Delete assets

AssetsAPI.**delete**(*id: Union[int, List[int]] = None*, *external_id: Union[str, List[str]] = None*, *recursive: bool = False*, *ignore_unknown_ids: bool = False*) → None

Delete one or more assets

Parameters

- **id** (*Union[int, List[int]]*) – Id or list of ids
- **external_id** (*Union[str, List[str]]*) – External ID or list of exgernal ids

- **recursive** (*bool*) – Recursively delete whole asset subtrees under given ids. Defaults to False.
- **ignore_unknown_ids** (*bool*) – Ignore IDs and external IDs that are not found rather than throw an exception.

Returns None

Examples

Delete assets by id or external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.assets.delete(id=[1,2,3], external_id="3")
```

Update assets

AssetsAPI.**update**(*item: Union[cognite.client.data_classes.assets.Asset, cognite.client.data_classes.assets.AssetUpdate, List[Union[cognite.client.data_classes.assets.Asset, cognite.client.data_classes.assets.AssetUpdate]]]*) → Union[cognite.client.data_classes.assets.Asset, cognite.client.data_classes.assets.AssetList]

Update one or more assets

Parameters **item** (*Union[Asset, AssetUpdate, List[Union[Asset, AssetUpdate]]]*) – Asset(s) to update

Returns Updated asset(s)

Return type Union[*Asset*, *AssetList*]

Examples

Update an asset that you have fetched. This will perform a full update of the asset:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> asset = c.assets.retrieve(id=1)
>>> asset.description = "New description"
>>> res = c.assets.update(asset)
```

Perform a partial update on a asset, updating the description and adding a new field to metadata:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import AssetUpdate
>>> c = CogniteClient()
>>> my_update = AssetUpdate(id=1).description.set("New description").metadata.add(
↪{"key": "value"})
>>> res = c.assets.update(my_update)
```

Data classes

class cognite.client.data_classes.assets.**AggregateResultItem**(*child_count: int = None, depth: int = None, path: List[Dict[str, Any]] = None, **kwargs*)

Bases: dict

Aggregated metrics of the asset

Parameters

- **child_count** (*int*) – Number of direct descendants for the asset
- **depth** (*int*) – Asset path depth (number of levels below root node).
- **path** (*List[Dict[str, Any]]*) – IDs of assets on the path to the asset.

class cognite.client.data_classes.assets.**Asset**(*external_id: str = None, name: str = None, parent_id: int = None, parent_external_id: str = None, description: str = None, data_set_id: int = None, metadata: Dict[str, str] = None, source: str = None, id: int = None, created_time: int = None, last_updated_time: int = None, root_id: int = None, aggregates: Union[Dict[str, Any], cognite.client.data_classes.assets.AggregateResultItem] = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

A representation of a physical asset, for example a factory or a piece of equipment.

Parameters

- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.
- **name** (*str*) – The name of the asset.
- **parent_id** (*int*) – The parent of the node, null if it is the root node.
- **parent_external_id** (*str*) – The external ID of the parent. The property is omitted if the asset doesn't have a parent or if the parent doesn't have externalId.
- **description** (*str*) – The description of the asset.
- **data_set_id** (*int*) – The id of the dataset this asset belongs to.
- **metadata** (*Dict[str, str]*) – Custom, application specific metadata. String key -> String value. Limits: Maximum length of key is 128 bytes, value 10240 bytes, up to 256 key-value pairs, of total size at most 10240.
- **source** (*str*) – The source of the asset.
- **id** (*int*) – A server-generated ID for the object.
- **created_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.

- **last_updated_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **root_id** (*int*) – ID of the root asset.
- **aggregates** (*Union[Dict[str, Any], AggregateResultItem]*) – Aggregated metrics of the asset
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

children() → cognite.client.data_classes.assets.AssetList
Returns the children of this asset.

Returns The requested assets

Return type *AssetList*

events(**kwargs) → EventList
Retrieve all events related to this asset.

Returns All events related to this asset.

Return type *EventList*

files(**kwargs) → FileMetadataList
Retrieve all files metadata related to this asset.

Returns Metadata about all files related to this asset.

Return type *FileMetadataList*

parent() → cognite.client.data_classes.assets.Asset
Returns this assets parent.

Returns The parent asset.

Return type *Asset*

sequences(**kwargs) → SequenceList
Retrieve all sequences related to this asset.

Returns All sequences related to this asset.

Return type *SequenceList*

subtree(*depth: int = None*) → cognite.client.data_classes.assets.AssetList
Returns the subtree of this asset up to a specified depth.

Parameters **depth** (*int, optional*) – Retrieve assets up to this depth below the asset.

Returns The requested assets sorted topologically.

Return type *AssetList*

time_series(**kwargs) → TimeSeriesList
Retrieve all time series related to this asset.

Returns All time series related to this asset.

Return type *TimeSeriesList*

to_pandas(*expand: List[str] = ('metadata', 'aggregates')*, *ignore: List[str] = None*, *camel_case: bool = True*)
Convert the instance into a pandas DataFrame.

Parameters

- **expand** (*List[str]*) – List of row keys to expand, only works if the value is a Dict.

- **ignore** (*List[str]*) – List of row keys to not include when converting to a data frame.
- **camel_case** (*bool*) – Convert column names to camel case (e.g. *externalId* instead of *external_id*)

Returns The dataframe.

Return type pandas.DataFrame

class cognite.client.data_classes.assets.**AssetAggregate**(*count: int = None*, ***kwargs*)

Bases: dict

Aggregation group of assets

Parameters **count** (*int*) – Size of the aggregation group

class cognite.client.data_classes.assets.**AssetFilter**(*name: str = None*, *parent_ids: List[int] = None*, *parent_external_ids: List[str] = None*, *root_ids: List[Dict[str, Any]] = None*, *asset_subtree_ids: List[Dict[str, Any]] = None*, *data_set_ids: List[Dict[str, Any]] = None*, *metadata: Dict[str, str] = None*, *source: str = None*, *created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None*, *last_updated_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None*, *root: bool = None*, *external_id_prefix: str = None*, *cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteFilter*

Filter on assets with strict matching.

Parameters

- **name** (*str*) – The name of the asset.
- **parent_ids** (*List[int]*) – Return only the direct descendants of the specified assets.
- **parent_external_ids** (*List[str]*) – Return only the direct descendants of the specified assets.
- **root_ids** (*List[Dict[str, Any]]*) – This parameter is deprecated. Use assetSubtreeIds instead. Only include these root assets and their descendants.
- **asset_subtree_ids** (*List[Dict[str, Any]]*) – Only include assets in subtrees rooted at the specified assets (including the roots given). If the total size of the given subtrees exceeds 100,000 assets, an error will be returned.
- **data_set_ids** (*List[Dict[str, Any]]*) – No description.
- **metadata** (*Dict[str, str]*) – Custom, application specific metadata. String key -> String value. Limits: Maximum length of key is 128 bytes, value 10240 bytes, up to 256 key-value pairs, of total size at most 10240.
- **source** (*str*) – The source of the asset.

- **created_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **last_updated_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **root** (*bool*) – Whether the filtered assets are root assets, or not. Set to True to only list root assets.
- **external_id_prefix** (*str*) – Filter by this (case-sensitive) prefix for the external ID.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.assets.**AssetList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

events() → EventList
Retrieve all events related to these assets.

Returns All events related to the assets in this AssetList.

Return type *EventList*

files() → FileMetadataList
Retrieve all files metadata related to these assets.

Returns Metadata about all files related to the assets in this AssetList.

Return type *FileMetadataList*

sequences() → SequenceList
Retrieve all sequences related to these assets.

Returns All sequences related to the assets in this AssetList.

Return type *SequenceList*

time_series() → TimeSeriesList
Retrieve all time series related to these assets.

Returns All time series related to the assets in this AssetList.

Return type *TimeSeriesList*

class cognite.client.data_classes.assets.**AssetUpdate**(*id: int = None, external_id: str = None*)

Bases: *cognite.client.data_classes._base.CogniteUpdate*

Changes applied to asset

Parameters

- **id** (*int*) – A server-generated ID for the object.
- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.

2.4.4 Events

Retrieve an event by id

EventsAPI.**retrieve**(*id: Optional[int] = None, external_id: Optional[str] = None*) → Optional[cognite.client.data_classes.events.Event]
Retrieve a single event by id.

Parameters

- **id** (*int, optional*) – ID
- **external_id** (*str, optional*) – External ID

Returns Requested event or None if it does not exist.

Return type Optional[*Event*]

Examples

Get event by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.events.retrieve(id=1)
```

Get event by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.events.retrieve(external_id="1")
```

Retrieve multiple events by id

EventsAPI.**retrieve_multiple**(*ids: Optional[List[int]] = None*, *external_ids: Optional[List[str]] = None*, *ignore_unknown_ids: bool = False*) → cognite.client.data_classes.events.EventList

Retrieve multiple events by id.

Parameters

- **ids** (*List[int], optional*) – IDs
- **external_ids** (*List[str], optional*) – External IDs
- **ignore_unknown_ids** (*bool*) – Ignore IDs and external IDs that are not found rather than throw an exception.

Returns The requested events.

Return type *EventList*

Examples

Get events by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.events.retrieve_multiple(ids=[1, 2, 3])
```

Get events by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.events.retrieve_multiple(external_ids=["abc", "def"])
```

List events

EventsAPI.**list**(*start_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, end_time: Union[Dict[str, Any], cognite.client.data_classes.events.EndTimeFilter] = None, active_at_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, type: str = None, subtype: str = None, metadata: Dict[str, str] = None, asset_ids: List[int] = None, asset_external_ids: List[str] = None, root_asset_ids: List[int] = None, root_asset_external_ids: List[str] = None, asset_subtree_ids: List[int] = None, asset_subtree_external_ids: List[str] = None, data_set_ids: List[int] = None, data_set_external_ids: List[str] = None, source: str = None, created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, last_updated_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, external_id_prefix: str = None, sort: List[str] = None, partitions: int = None, limit: int = 25*) → cognite.client.data_classes.events.EventList

List events

Parameters

- **start_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **end_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **type** (*str*) – Type of the event, e.g ‘failure’.
- **subtype** (*str*) – Subtype of the event, e.g ‘electrical’.
- **metadata** (*Dict[str, str]*) – Customizable extra data about the event. String key -> String value.
- **asset_ids** (*List[int]*) – Asset IDs of related equipments that this event relates to.
- **asset_external_ids** (*List[str]*) – Asset External IDs of related equipment that this event relates to.
- **root_asset_ids** (*List[int]*) – The IDs of the root assets that the related assets should be children of.
- **root_asset_external_ids** (*List[str]*) – The external IDs of the root assets that the related assets should be children of.
- **asset_subtree_ids** (*List[int]*) – List of asset subtrees ids to filter on.
- **asset_subtree_external_ids** (*List[str]*) – List of asset subtrees external ids to filter on.
- **data_set_ids** (*List[int]*) – Return only events in the specified data sets with these ids.
- **data_set_external_ids** (*List[str]*) – Return only events in the specified data sets with these external ids.
- **source** (*str*) – The source of this event.
- **created_time** (*Union[Dict[str, int], TimestampRange]*) – Range between two timestamps. Possible keys are *min* and *max*, with values given as time stamps in ms.

- **last_updated_time** (*Union[Dict[str, int], TimestampRange]*) – Range between two timestamps. Possible keys are *min* and *max*, with values given as time stamps in ms.
- **external_id_prefix** (*str*) – External Id provided by client. Should be unique within the project.
- **sort** (*List[str]*) – Sort by array of selected fields. Ex: [“startTime:desc’]. Default sort order is asc when ommitted. Filter accepts following field names: startTime, endTime, createdTime, lastUpdatedTime. We only support 1 field for now.
- **partitions** (*int*) – Retrieve events in parallel using this number of workers. Also requires *limit=None* to be passed.
- **limit** (*int, optional*) – Maximum number of events to return. Defaults to 25. Set to -1, float(“inf”) or None to return all items.

Returns List of requested events

Return type *EventList*

Examples

List events and filter on max start time:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> event_list = c.events.list(limit=5, start_time={"max": 1500000000})
```

Iterate over events:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for event in c.events:
...     event # do something with the event
```

Iterate over chunks of events to reduce memory load:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for event_list in c.events(chunk_size=2500):
...     event_list # do something with the events
```

Aggregate events

EventsAPI.**aggregate**(*filter: Union[cognite.client.data_classes.events.EventFilter, Dict[KT, VT]] = None*) → List[cognite.client.data_classes.events.EventAggregate]

Aggregate events

Parameters **filter** (*Union[EventFilter, Dict]*) – Filter on events filter with exact match

Returns List of event aggregates

Return type List[*EventAggregate*]

Examples

Aggregate events:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> aggregate_type = c.events.aggregate(filter={"type": "failure"})
```

Search for events

EventsAPI.**search**(*description: str = None*, *filter: Union[cognite.client.data_classes.events.EventFilter, Dict[KT, VT]] = None*, *limit: int = 100*) → cognite.client.data_classes.events.EventList

Search for events Primarily meant for human-centric use-cases and data exploration, not for programs, since matching and ordering may change over time. Use the *list* function if stable or exact matches are required.

Parameters

- **description** (*str*) – Fuzzy match on description.
- **filter** (*Union[EventFilter, Dict]*) – Filter to apply. Performs exact match on these fields.
- **limit** (*int*) – Maximum number of results to return.

Returns List of requested events

Return type *EventList*

Examples

Search for events:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.events.search(description="some description")
```

Create events

EventsAPI.**create**(*event: Union[cognite.client.data_classes.events.Event, List[cognite.client.data_classes.events.Event]]*) → Union[cognite.client.data_classes.events.Event, cognite.client.data_classes.events.EventList]

Create one or more events.

Parameters **event** (*Union[Event, List[Event]]*) – Event or list of events to create.

Returns Created event(s)

Return type Union[*Event*, *EventList*]

Examples

Create new events:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import Event
>>> c = CogniteClient()
>>> events = [Event(start_time=0, end_time=1), Event(start_time=2, end_time=3)]
>>> res = c.events.create(events)
```

Delete events

EventsAPI.**delete**(*id: Union[int, List[int]] = None, external_id: Union[str, List[str]] = None, ignore_unknown_ids: bool = False*) → None

Delete one or more events

Parameters

- **id** (*Union[int, List[int]]*) – Id or list of ids
- **external_id** (*Union[str, List[str]]*) – External ID or list of external ids
- **ignore_unknown_ids** (*bool*) – Ignore IDs and external IDs that are not found rather than throw an exception.

Returns None

Examples

Delete events by id or external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.events.delete(id=[1,2,3], external_id="3")
```

Update events

EventsAPI.**update**(*item: Union[cognite.client.data_classes.events.Event, cognite.client.data_classes.events.EventUpdate, List[Union[cognite.client.data_classes.events.Event, cognite.client.data_classes.events.EventUpdate]]]*) → Union[cognite.client.data_classes.events.Event, cognite.client.data_classes.events.EventList]

Update one or more events

Parameters **item** (*Union[Event, EventUpdate, List[Union[Event, EventUpdate]]]*) – Event(s) to update

Returns Updated event(s)

Return type Union[*Event*, *EventList*]

Examples

Update an event that you have fetched. This will perform a full update of the event:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> event = c.events.retrieve(id=1)
```

(continues on next page)

(continued from previous page)

```
>>> event.description = "New description"
>>> res = c.events.update(event)
```

Perform a partial update on a event, updating the description and adding a new field to metadata:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import EventUpdate
>>> c = CogniteClient()
>>> my_update = EventUpdate(id=1).description.set("New description").metadata.add(
↪{"key": "value"})
>>> res = c.events.update(my_update)
```

Data classes

class cognite.client.data_classes.events.**EndTimeFilter**(*max: int = None*, *min: int = None*, *is_null: bool = None*, ***kwargs*)

Bases: `dict`

Either range between two timestamps or isNull filter condition.

Parameters

- **max** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **min** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **is_null** (*bool*) – Set to true if you want to search for data with field value not set, false to search for cases where some value is present.

class cognite.client.data_classes.events.**Event**(*external_id: str = None*, *data_set_id: int = None*, *start_time: int = None*, *end_time: int = None*, *type: str = None*, *subtype: str = None*, *description: str = None*, *metadata: Dict[str, str] = None*, *asset_ids: List[int] = None*, *source: str = None*, *id: int = None*, *last_updated_time: int = None*, *created_time: int = None*, *cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

An event represents something that happened at a given interval in time, e.g a failure, a work order etc.

Parameters

- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.
- **data_set_id** (*int*) – The id of the dataset this event belongs to.
- **start_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **end_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.

- **type** (*str*) – Type of the event, e.g 'failure'.
- **subtype** (*str*) – Subtype of the event, e.g 'electrical'.
- **description** (*str*) – Textual description of the event.
- **metadata** (*Dict[str, str]*) – Custom, application specific metadata. String key -> String value. Limits: Maximum length of key is 128 bytes, value 128000 bytes, up to 256 key-value pairs, of total size at most 200000.
- **asset_ids** (*List[int]*) – Asset IDs of equipment that this event relates to.
- **source** (*str*) – The source of this event.
- **id** (*int*) – A server-generated ID for the object.
- **last_updated_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **created_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.events.**EventAggregate**(*count: int = None*, ***kwargs*)

Bases: dict

Aggregation group of events

Parameters **count** (*int*) – Size of the aggregation group

class cognite.client.data_classes.events.**EventFilter**(*start_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, end_time: Union[Dict[str, Any], cognite.client.data_classes.events.EndTimeFilter] = None, active_at_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, metadata: Dict[str, str] = None, asset_ids: List[int] = None, asset_external_ids: List[str] = None, root_asset_ids: List[Dict[str, Any]] = None, asset_subtree_ids: List[Dict[str, Any]] = None, data_set_ids: List[Dict[str, Any]] = None, source: str = None, type: str = None, subtype: str = None, created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, last_updated_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, external_id_prefix: str = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteFilter*

Filter on events filter with exact match

Parameters

- **start_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **end_time** (*Union[Dict[str, Any], EndTimeFilter]*) – Either range between two timestamps or isNull filter condition.
- **active_at_time** (*Union[Dict[str, Any], TimestampRange]*) – Event is considered active from its startTime to endTime inclusive. If startTime is null, event is never active. If endTime is null, event is active from startTime onwards. activeAtTime filter will match all events that are active at some point from min to max, from min, or to max, depending on which of min and max parameters are specified.
- **metadata** (*Dict[str, str]*) – Custom, application specific metadata. String key -> String value. Limits: Maximum length of key is 128 bytes, value 128000 bytes, up to 256 key-value pairs, of total size at most 200000.
- **asset_ids** (*List[int]*) – Asset IDs of equipment that this event relates to.
- **asset_external_ids** (*List[str]*) – Asset External IDs of equipment that this event relates to.
- **root_asset_ids** (*List[Dict[str, Any]]*) – This parameter is deprecated. Use assetSubtreeIds instead. Only include events that have a related asset in a tree rooted at any of these root assetIds.
- **asset_subtree_ids** (*List[Dict[str, Any]]*) – Only include events that have a related asset in a subtree rooted at any of these assetIds (including the roots given). If the total size of the given subtrees exceeds 100,000 assets, an error will be returned.
- **data_set_ids** (*List[Dict[str, Any]]*) – Only include events that belong to these datasets.
- **source** (*str*) – The source of this event.
- **type** (*str*) – The event type
- **subtype** (*str*) – The event subtype
- **created_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **last_updated_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **external_id_prefix** (*str*) – Filter by this (case-sensitive) prefix for the external ID.
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

class cognite.client.data_classes.events.**EventList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.events.**EventUpdate**(*id: int = None, external_id: str = None*)

Bases: *cognite.client.data_classes._base.CogniteUpdate*

Changes will be applied to event.

Parameters

- **id** (*int*) – A server-generated ID for the object.

- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.

2.4.5 Data sets

Retrieve an data set by id

DataSetsAPI.**retrieve**(*id: Optional[int] = None, external_id: Optional[str] = None*) → Optional[cognite.client.data_classes.data_sets.DataSet]

Retrieve a single data set by id.

Parameters

- **id** (*int, optional*) – ID
- **external_id** (*str, optional*) – External ID

Returns Requested data set or None if it does not exist.

Return type Optional[*DataSet*]

Examples

Get data set by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.data_sets.retrieve(id=1)
```

Get data set by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.data_sets.retrieve(external_id="1")
```

Retrieve multiple data sets by id

DataSetsAPI.**retrieve_multiple**(*ids: Optional[List[int]] = None, external_ids: Optional[List[str]] = None, ignore_unknown_ids: bool = False*) → cognite.client.data_classes.data_sets.DataSetList

Retrieve multiple data sets by id.

Parameters

- **ids** (*List[int], optional*) – IDs
- **external_ids** (*List[str], optional*) – External IDs
- **ignore_unknown_ids** (*bool*) – Ignore IDs and external IDs that are not found rather than throw an exception.

Returns The requested data sets.

Return type *DataSetList*

Examples

Get data sets by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.data_sets.retrieve_multiple(ids=[1, 2, 3])
```

Get data sets by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.data_sets.retrieve_multiple(external_ids=["abc", "def"], ignore_
↪unknown_ids=True)
```

List data sets

DataSetsAPI.**list**(*metadata: Dict[str, str] = None, created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, last_updated_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, external_id_prefix: str = None, write_protected: bool = None, limit: int = 25*) → cognite.client.data_classes.data_sets.DataSetList

List data sets

Parameters

- **metadata** (*Dict[str, str]*) – Custom, application-specific metadata. String key -> String value.
- **created_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **last_updated_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **external_id_prefix** (*str*) – Filter by this (case-sensitive) prefix for the external ID.
- **write_protected** (*bool*) – Specify whether the filtered data sets are write-protected, or not. Set to True to only list write-protected data sets.
- **limit** (*int, optional*) – Maximum number of data sets to return. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns List of requested data sets

Return type *DataSetList*

Examples

List data sets and filter on write_protected:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> data_sets_list = c.data_sets.list(limit=5, write_protected=False)
```

Iterate over data sets:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for data_set in c.data_sets:
...     data_set # do something with the data_set
```

Iterate over chunks of data sets to reduce memory load:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for data_set_list in c.data_sets(chunk_size=2500):
...     data_set_list # do something with the list
```

Aggregate data sets

DataSetsAPI.**aggregate**(*filter: Union[cognite.client.data_classes.data_sets.DataSetFilter, Dict[KT, VT]] = None*) → List[cognite.client.data_classes.data_sets.DataSetAggregate]

Aggregate data sets

Parameters **filter** (*Union[DataSetFilter, Dict]*) – Filter on data set filter with exact match

Returns List of data set aggregates

Return type List[*DataSetAggregate*]

Examples

Aggregate data_sets:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> aggregate_protected = c.data_sets.aggregate(filter={"write_protected": True})
```

Create data sets

DataSetsAPI.**create**(*data_set: Union[cognite.client.data_classes.data_sets.DataSet, List[cognite.client.data_classes.data_sets.DataSet]]*) → Union[cognite.client.data_classes.data_sets.DataSet, cognite.client.data_classes.data_sets.DataSetList]

Create one or more data sets.

Parameters **data_set** – Union[DataSet, List[DataSet]]: Data set or list of data sets to create.

Returns Created data set(s)

Return type Union[*DataSet*, *DataSetList*]

Examples

Create new data sets:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import DataSet
>>> c = CogniteClient()
>>> data_sets = [DataSet(name="1st level"), DataSet(name="2nd level")]
>>> res = c.data_sets.create(data_sets)
```

Delete data sets

This functionality is not yet available in the API.

Update data sets

DataSetsAPI.**update**(*item: Union[cognite.client.data_classes.data_sets.DataSet, cognite.client.data_classes.data_sets.DataSetUpdate, List[Union[cognite.client.data_classes.data_sets.DataSet, cognite.client.data_classes.data_sets.DataSetUpdate]]]*) → Union[cognite.client.data_classes.data_sets.DataSet, cognite.client.data_classes.data_sets.DataSetList]

Update one or more data sets

Parameters **item** – Union[DataSet, DataSetUpdate, List[Union[DataSet, DataSetUpdate]]]: Data set(s) to update

Returns Updated data set(s)

Return type Union[*DataSet*, *DataSetList*]

Examples

Update a data set that you have fetched. This will perform a full update of the data set:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> data_set = c.data_sets.retrieve(id=1)
>>> data_set.description = "New description"
>>> res = c.data_sets.update(data_set)
```

Perform a partial update on a data set, updating the description and removing a field from metadata:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import DataSetUpdate
>>> c = CogniteClient()
>>> my_update = DataSetUpdate(id=1).description.set("New description").metadata.
↪remove(["key"])
>>> res = c.data_sets.update(my_update)
```

Data classes

class cognite.client.data_classes.data_sets.**DataSet**(*external_id: str = None*, *name: str = None*, *description: str = None*, *metadata: Dict[str, str] = None*, *write_protected: bool = None*, *id: int = None*, *created_time: int = None*, *last_updated_time: int = None*, *cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

No description.

Parameters

- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.
- **name** (*str*) – The name of the data set.
- **description** (*str*) – The description of the data set.
- **metadata** (*Dict[str, str]*) – Custom, application-specific metadata. String key -> String value.
- **write_protected** (*bool*) – To write data to a write-protected data set, you need to be a member of a group that has the "datasets:owner" action for the data set. To learn more about write-protected data sets, follow this [guide](/cdf/data_governance/concepts/datasets/#write-protection)
- **id** (*int*) – A server-generated ID for the object.
- **created_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **last_updated_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.data_sets.**DataSetAggregate**(*count: int = None*, ***kwargs*)

Bases: dict

Aggregation group of data sets

Parameters **count** (*int*) – Size of the aggregation group

class cognite.client.data_classes.data_sets.**DataSetFilter**(*metadata: Dict[str, str] = None, created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange = None, last_updated_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange = None, external_id_prefix: str = None, write_protected: bool = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteFilter*

Filter on data sets with strict matching.

Parameters

- **metadata** (*Dict[str, str]*) – Custom, application-specific metadata. String key -> String value.
- **created_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **last_updated_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **external_id_prefix** (*str*) – Filter by this (case-sensitive) prefix for the external ID.
- **write_protected** (*bool*) – No description.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.data_sets.**DataSetList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.data_sets.**DataSetUpdate**(*id: int = None, external_id: str = None*)

Bases: *cognite.client.data_classes._base.CogniteUpdate*

Update applied to single data set

Parameters

- **id** (*int*) – A server-generated ID for the object.
- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.

2.4.6 Files

Retrieve file metadata by id

FilesAPI.**retrieve**(*id: Optional[int] = None, external_id: Optional[str] = None*) → Optional[cognite.client.data_classes.files.FileMetadata]

Retrieve a single file metadata by id.

Parameters

- **id** (*int, optional*) – ID
- **external_id** (*str, optional*) – External ID

Returns Requested file metadata or None if it does not exist.

Return type Optional[*FileMetadata*]

Examples

Get file metadata by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.files.retrieve(id=1)
```

Get file metadata by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.files.retrieve(external_id="1")
```

Retrieve multiple files' metadata by id

FilesAPI.**retrieve_multiple**(*ids: Optional[List[int]] = None*, *external_ids: Optional[List[str]] = None*) → cognite.client.data_classes.files.FileMetadataList

Retrieve multiple file metadatas by id.

Parameters

- **ids** (*List[int], optional*) – IDs
- **external_ids** (*List[str], optional*) – External IDs

Returns The requested file metadatas.

Return type *FileMetadataList*

Examples

Get file metadatas by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.files.retrieve_multiple(ids=[1, 2, 3])
```

Get file_metadatas by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.files.retrieve_multiple(external_ids=["abc", "def"])
```

List files metadata

FilesAPI.**list**(*name: str = None, mime_type: str = None, metadata: Dict[str, str] = None, asset_ids: List[int] = None, asset_external_ids: List[str] = None, root_asset_ids: List[int] = None, root_asset_external_ids: List[str] = None, asset_subtree_ids: List[int] = None, asset_subtree_external_ids: List[str] = None, data_set_ids: List[int] = None, data_set_external_ids: List[str] = None, source: str = None, created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, last_updated_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, source_created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, source_modified_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, uploaded_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, external_id_prefix: str = None, uploaded: bool = None, limit: int = 25*) → cognite.client.data_classes.files.FileMetadataList

List files

Parameters

- **name** (*str*) – Name of the file.
- **mime_type** (*str*) – File type. E.g. text/plain, application/pdf, ..
- **metadata** (*Dict[str, str]*) – Custom, application specific metadata. String key -> String value
- **asset_ids** (*List[int]*) – Only include files that reference these specific asset IDs.
- **asset_subtree_external_ids** (*List[str]*) – Only include files that reference these specific asset external IDs.
- **root_asset_ids** (*List[int]*) – The IDs of the root assets that the related assets should be children of.
- **root_asset_external_ids** (*List[str]*) – The external IDs of the root assets that the related assets should be children of.
- **asset_subtree_ids** (*List[int]*) – List of asset subtrees ids to filter on.
- **asset_subtree_external_ids** – List of asset subtrees external ids to filter on.
- **data_set_ids** (*List[int]*) – Return only files in the specified data sets with these ids.
- **data_set_external_ids** (*List[str]*) – Return only files in the specified data sets with these external ids.
- **source** (*str*) – The source of this event.
- **created_time** (*Union[Dict[str, int], TimestampRange]*) – Range between two timestamps. Possible keys are *min* and *max*, with values given as time stamps in ms.
- **last_updated_time** (*Union[Dict[str, int], TimestampRange]*) – Range between two timestamps. Possible keys are *min* and *max*, with values given as time stamps in ms.
- **uploaded_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps

- **source_created_time** (*Union[Dict[str, Any], TimestampRange]*) – Filter for files where the sourceCreatedTime field has been set and is within the specified range.
- **source_modified_time** (*Union[Dict[str, Any], TimestampRange]*) – Filter for files where the sourceModifiedTime field has been set and is within the specified range.
- **external_id_prefix** (*str*) – External Id provided by client. Should be unique within the project.
- **uploaded** (*bool*) – Whether or not the actual file is uploaded. This field is returned only by the API, it has no effect in a post body.
- **limit** (*int, optional*) – Max number of files to return. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns The requested files.

Return type *FileMetadataList*

Examples

List files metadata and filter on external id prefix:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> file_list = c.files.list(limit=5, external_id_prefix="prefix")
```

Iterate over files metadata:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for file_metadata in c.files:
...     file_metadata # do something with the file metadata
```

Iterate over chunks of files metadata to reduce memory load:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for file_list in c.files(chunk_size=2500):
...     file_list # do something with the files
```

Aggregate files metadata

FilesAPI.**aggregate**(*filter: Union[cognite.client.data_classes.files.FileMetadataFilter, Dict[KT, VT]] = None*) → List[cognite.client.data_classes.files.FileAggregate]

Aggregate files

Parameters **filter** (*Union[FileMetadataFilter, Dict]*) – Filter on file metadata filter with exact match

Returns List of file aggregates

Return type List[*FileAggregate*]

Examples

List files metadata and filter on external id prefix:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> aggregate_uploaded = c.files.aggregate(filter={"uploaded": True})
```

Search for files

FilesAPI.**search**(*name: str = None*, *filter: Union[cognite.client.data_classes.files.FileMetadataFilter, dict] = None*, *limit: int = 100*) → cognite.client.data_classes.files.FileMetadataList

Search for files. Primarily meant for human-centric use-cases and data exploration, not for programs, since matching and ordering may change over time. Use the *list* function if stable or exact matches are required.

Parameters

- **name** (*str, optional*) – Prefix and fuzzy search on name.
- **filter** (*Union[FileMetadataFilter, dict], optional*) – Filter to apply. Performs exact match on these fields.
- **limit** (*int, optional*) – Max number of results to return.

Returns List of requested files metadata.

Return type *FileMetadataList*

Examples

Search for a file:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.files.search(name="some name")
```

Create file metadata

FilesAPI.**create**(*file_metadata: cognite.client.data_classes.files.FileMetadata*, *overwrite: bool = False*) → Tuple[cognite.client.data_classes.files.FileMetadata, str]

Create file without uploading content.

Parameters

- **file_metadata** (*FileMetaData*) – File metadata for the file to create.
- **overwrite** (*bool*) – If 'overwrite' is set to true, and the POST body content specifies a 'externalId' field, fields for the file found for externalId can be overwritten. The default setting is false. If metadata is included in the request body, all of the original metadata will be overwritten. File-Asset mappings only change if explicitly stated in the assetIds field of the POST json body. Do not set assetIds in request body if you want to keep the current file-asset mappings.

Returns Tuple containing the file metadata and upload url of the created file.

Return type Tuple[FileMetaData, str]

Examples

Create a file:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import FileMetadata
>>> c = CogniteClient()
>>> file_metadata = FileMetadata(name="MyFile")
>>> res = c.files.create(file_metadata)
```

Upload a file or directory

FilesAPI.**upload**(*path: str*, *external_id: str = None*, *name: str = None*, *source: str = None*, *mime_type: str = None*, *metadata: Dict[str, str] = None*, *asset_ids: List[int] = None*, *source_created_time: int = None*, *source_modified_time: int = None*, *data_set_id: int = None*, *security_categories: List[int] = None*, *recursive: bool = False*, *overwrite: bool = False*) → Union[cognite.client.data_classes.files.FileMetadata, cognite.client.data_classes.files.FileMetadataList]

Upload a file

Parameters

- **path** (*str*) – Path to the file you wish to upload. If path is a directory, this method will upload all files in that directory.
- **external_id** (*str*) – The external ID provided by the client. Must be unique within the project.
- **name** (*str*) – Name of the file.
- **source** (*str*) – The source of the file.
- **mime_type** (*str*) – File type. E.g. text/plain, application/pdf, ...
- **metadata** (*Dict[str, str]*) – Customizable extra data about the file. String key -> String value.
- **asset_ids** (*List[int]*) – No description.
- **data_set_id** (*int*) – ID of the data set.
- **source_created_time** (*int*) – The timestamp for when the file was originally created in the source system.
- **source_modified_time** (*int*) – The timestamp for when the file was last modified in the source system.
- **recursive** (*bool*) – If path is a directory, upload all contained files recursively.
- **overwrite** (*bool*) – If 'overwrite' is set to true, and the POST body content specifies a 'externalId' field, fields for the file found for externalId can be overwritten. The default setting is false. If metadata is included in the request body, all of the original metadata will be overwritten. The actual file will be overwritten after successful upload. If there is no successful upload, the current file contents will be kept. File-Asset mappings only change if explicitly stated in the assetIds field of the POST json body. Do not set assetIds in request body if you want to keep the current file-asset mappings.

Returns The file metadata of the uploaded file(s).

Return type Union[*FileMetadata*, *FileMetadataList*]

Examples

Upload a file in a given path:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.files.upload("/path/to/file", name="my_file")
```

If name is omitted, this method will use the name of the file

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.files.upload("/path/to/file")
```

You can also upload all files in a directory by setting path to the path of a directory:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.files.upload("/path/to/my/directory")
```

Upload a string or bytes

FilesAPI.**upload_bytes**(*content: Union[str, bytes, TextIO, BinaryIO], name: str, external_id: str = None, source: str = None, mime_type: str = None, metadata: Dict[str, str] = None, asset_ids: List[int] = None, data_set_id: int = None, source_created_time: int = None, source_modified_time: int = None, security_categories: List[int] = None, overwrite: bool = False*)

Upload bytes or string.

You can also pass a file handle to content.

Parameters

- **content** (*Union[str, bytes, TextIO, BinaryIO]*) – The content to upload.
- **name** (*str*) – Name of the file.
- **external_id** (*str*) – The external ID provided by the client. Must be unique within the project.
- **source** (*str*) – The source of the file.
- **mime_type** (*str*) – File type. E.g. text/plain, application/pdf,...
- **metadata** (*Dict[str, str]*) – Customizable extra data about the file. String key -> String value.
- **asset_ids** (*List[int]*) – No description.
- **data_set_id** (*int*) – Id of the data set.
- **source_created_time** (*int*) – The timestamp for when the file was originally created in the source system.
- **source_modified_time** (*int*) – The timestamp for when the file was last modified in the source system.
- **overwrite** (*bool*) – If 'overwrite' is set to true, and the POST body content specifies a 'externalId' field, fields for the file found for externalId can be overwritten. The default setting is false. If metadata is included in the request body, all of the original metadata will be overwritten. The actual file will be overwritten after successful upload. If there is no

successful upload, the current file contents will be kept. File-Asset mappings only change if explicitly stated in the assetIds field of the POST json body. Do not set assetIds in request body if you want to keep the current file-asset mappings.

Examples

Upload a file from memory:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.files.upload_bytes(b"some content", name="my_file", asset_ids=[1,2,3])
```

Download files to disk

FilesAPI.**download**(*directory: str*, *id: Union[int, List[int]] = None*, *external_id: Union[str, List[str]] = None*) → None

Download files by id or external id.

This method will stream all files to disk, never keeping more than 2MB of a given file in memory.

Parameters

- **directory** (*str*) – Directory to download the file(s) to.
- **id** (*Union[int, List[int]], optional*) – Id or list of ids
- **external_id** (*Union[str, List[str]), optional*) – External ID or list of external ids.

Returns None

Examples

Download files by id and external id into directory 'my_directory':

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.files.download(directory="my_directory", id=[1,2,3], external_id=["abc",
↪"def"])
```

Download a single file to a specific path

FilesAPI.**download_to_path**(*path: str*, *id: int = None*, *external_id: str = None*)

Download a file to a specific target.

Parameters

- **path** (*str*) – The path in which to place the file.
- **id** (*int*) – Id of of the file to download.
- **external_id** (*str*) – External id of the file to download.

Returns None

Examples

Download a file by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.files.download_to_path("~/mydir/my_downloaded_file.txt", id=123)
```

Download a file as bytes

`FilesAPI.`**`download_bytes`**(*id: int = None*, *external_id: str = None*) → bytes

Download a file as bytes.

Parameters

- **id** (*int, optional*) – Id of the file
- **external_id** (*str, optional*) – External id of the file

Examples

Download a file's content into memory:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> file_content = c.files.download_bytes(id=1)
```

Delete files

`FilesAPI.`**`delete`**(*id: Union[int, List[int]] = None*, *external_id: Union[str, List[str]] = None*) → None

Delete files

Parameters

- **id** (*Union[int, List[int]]*) – Id or list of ids
- **external_id** (*Union[str, List[str]]*) – str or list of str

Returns None

Examples

Delete files by id or external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.files.delete(id=[1,2,3], external_id="3")
```

Update files metadata

FilesAPI.**update**(*item: Union[cognite.client.data_classes.files.FileMetadata, cognite.client.data_classes.files.FileMetadataUpdate, List[Union[cognite.client.data_classes.files.FileMetadata, cognite.client.data_classes.files.FileMetadataUpdate]]]*) → Union[cognite.client.data_classes.files.FileMetadata, cognite.client.data_classes.files.FileMetadataList]

Update files

Parameters **item** (*Union[FileMetadata, FileMetadataUpdate, List[Union[FileMetadata, FileMetadataUpdate]]]*) – file(s) to update.

Returns The updated files.

Return type Union[*FileMetadata*, *FileMetadataList*]

Examples

Update file metadata that you have fetched. This will perform a full update of the file metadata:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> file_metadata = c.files.retrieve(id=1)
>>> file_metadata.description = "New description"
>>> res = c.files.update(file_metadata)
```

Perform a partial update on file metadata, updating the source and adding a new field to metadata:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import FileMetadataUpdate
>>> c = CogniteClient()
>>> my_update = FileMetadataUpdate(id=1).source.set("new source").metadata.add({
↪"key": "value"})
>>> res = c.files.update(my_update)
```

Data classes

class cognite.client.data_classes.files.**FileAggregate**(*count: int = None*, ***kwargs*)

Bases: dict

Aggregation results for files

Parameters **count** (*int*) – Number of filtered items included in aggregation

class cognite.client.data_classes.files.**FileMetadata**(*external_id: str = None*, *name: str = None*, *source: str = None*, *mime_type: str = None*, *metadata: Dict[str, str] = None*, *asset_ids: List[int] = None*, *data_set_id: int = None*, *source_created_time: int = None*, *source_modified_time: int = None*, *security_categories: List[int] = None*, *id: int = None*, *uploaded: bool = None*, *uploaded_time: int = None*, *created_time: int = None*, *last_updated_time: int = None*, *cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

No description.

Parameters

- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.
- **name** (*str*) – Name of the file.
- **source** (*str*) – The source of the file.
- **mime_type** (*str*) – File type. E.g. text/plain, application/pdf, ..
- **metadata** (*Dict[str, str]*) – Custom, application specific metadata. String key -> String value. Limits: Maximum length of key is 32 bytes, value 512 bytes, up to 16 key-value pairs.
- **asset_ids** (*List[int]*) – No description.
- **data_set_id** (*int*) – The dataSet Id for the item.
- **source_created_time** (*int*) – The timestamp for when the file was originally created in the source system.
- **source_modified_time** (*int*) – The timestamp for when the file was last modified in the source system.
- **security_categories** (*List[int]*) – The security category IDs required to access this file.
- **id** (*int*) – A server-generated ID for the object.
- **uploaded** (*bool*) – Whether or not the actual file is uploaded. This field is returned only by the API, it has no effect in a post body.
- **uploaded_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **created_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **last_updated_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

class cognite.client.data_classes.files.**FileMetadataFilter**(*name: str = None, mime_type: str = None, metadata: Dict[str, str] = None, asset_ids: List[int] = None, asset_external_ids: List[str] = None, root_asset_ids: List[Dict[str, Any]] = None, data_set_ids: List[Dict[str, Any]] = None, asset_subtree_ids: List[Dict[str, Any]] = None, source: str = None, created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRang = None, last_updated_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRang = None, uploaded_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRang = None, source_created_time: Dict[str, Any] = None, source_modified_time: Dict[str, Any] = None, external_id_prefix: str = None, uploaded: bool = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteFilter*

No description.

Parameters

- **name** (*str*) – Name of the file.
- **mime_type** (*str*) – File type. E.g. text/plain, application/pdf, ..
- **metadata** (*Dict[str, str]*) – Custom, application specific metadata. String key -> String value. Limits: Maximum length of key is 32 bytes, value 512 bytes, up to 16 key-value pairs.
- **asset_ids** (*List[int]*) – Only include files that reference these specific asset IDs.
- **asset_external_ids** (*List[str]*) – Only include files that reference these specific asset external IDs.

- **root_asset_ids** (*List[Dict[str, Any]]*) – Only include files that have a related asset in a tree rooted at any of these root assetIds.
- **data_set_ids** (*List[Dict[str, Any]]*) – Only include files that belong to these datasets.
- **asset_subtree_ids** (*List[Dict[str, Any]]*) – Only include files that have a related asset in a subtree rooted at any of these assetIds (including the roots given). If the total size of the given subtrees exceeds 100,000 assets, an error will be returned.
- **source** (*str*) – The source of this event.
- **created_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **last_updated_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **uploaded_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **source_created_time** (*Dict[str, Any]*) – Filter for files where the sourceCreatedTime field has been set and is within the specified range.
- **source_modified_time** (*Dict[str, Any]*) – Filter for files where the sourceModifiedTime field has been set and is within the specified range.
- **external_id_prefix** (*str*) – Filter by this (case-sensitive) prefix for the external ID.
- **uploaded** (*bool*) – Whether or not the actual file is uploaded. This field is returned only by the API, it has no effect in a post body.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.files.**FileMetadataList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.files.**FileMetadataUpdate**(*id: int = None, external_id: str = None*)

Bases: *cognite.client.data_classes._base.CogniteUpdate*

Changes will be applied to file.

Args:

2.4.7 Time series

Retrieve a time series by id

TimeSeriesAPI.**retrieve**(*id: Optional[int] = None, external_id: Optional[str] = None*) → Optional[cognite.client.data_classes.time_series.TimeSeries]

Retrieve a single time series by id.

Parameters

- **id** (*int, optional*) – ID
- **external_id** (*str, optional*) – External ID

Returns Requested time series or None if it does not exist.

Return type Optional[*TimeSeries*]

Examples

Get time series by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.time_series.retrieve(id=1)
```

Get time series by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.time_series.retrieve(external_id="1")
```

Retrieve multiple time series by id

TimeSeriesAPI.**retrieve_multiple**(*ids: Optional[List[int]] = None*, *external_ids: Optional[List[str]] = None*, *ignore_unknown_ids: bool = False*) → cognite.client.data_classes.time_series.TimeSeriesList

Retrieve multiple time series by id.

Parameters

- **ids** (*List[int], optional*) – IDs
- **external_ids** (*List[str], optional*) – External IDs
- **ignore_unknown_ids** (*bool*) – Ignore IDs and external IDs that are not found rather than throw an exception.

Returns The requested time series.

Return type *TimeSeriesList*

Examples

Get time series by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.time_series.retrieve_multiple(ids=[1, 2, 3])
```

Get time series by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.time_series.retrieve_multiple(external_ids=["abc", "def"])
```

List time series

TimeSeriesAPI.**list**(*name: str = None*, *unit: str = None*, *is_string: bool = None*, *is_step: bool = None*, *asset_ids: List[int] = None*, *asset_external_ids: List[str] = None*, *root_asset_ids: List[int] = None*, *asset_subtree_ids: List[int] = None*, *asset_subtree_external_ids: List[str] = None*, *data_set_ids: List[int] = None*, *data_set_external_ids: List[str] = None*, *metadata: Dict[str, Any] = None*, *external_id_prefix: str = None*, *created_time: Dict[str, Any] = None*, *last_updated_time: Dict[str, Any] = None*, *partitions: int = None*, *limit: int = 25*, *include_metadata=True*) → cognite.client.data_classes.time_series.TimeSeriesList

List over time series

Fetches time series as they are iterated over, so you keep a limited number of objects in memory.

Parameters

- **name** (*str*) – Name of the time series. Often referred to as tag.
- **unit** (*str*) – Unit of the time series.
- **is_string** (*bool*) – Whether the time series is an string time series.
- **is_step** (*bool*) – Whether the time series is a step (piecewise constant) time series.
- **asset_ids** (*List[int], optional*) – List time series related to these assets.
- **asset_external_ids** (*List[str], optional*) – List time series related to these assets.
- **root_asset_ids** (*List[int], optional*) – List time series related to assets under these root assets.
- **asset_subtree_ids** (*List[int]*) – List of asset subtrees ids to filter on.
- **asset_subtree_external_ids** (*List[str]*) – List of asset subtrees external ids to filter on.
- **data_set_ids** (*List[int]*) – Return only assets in the specified data sets with these ids.
- **data_set_external_ids** (*List[str]*) – Return only assets in the specified data sets with these external ids.
- **metadata** (*Dict[str, Any]*) – Custom, application specific metadata. String key -> String value
- **created_time** (*Union[Dict[str, int], TimestampRange]*) – Range between two timestamps. Possible keys are *min* and *max*, with values given as time stamps in ms.
- **last_updated_time** (*Union[Dict[str, int], TimestampRange]*) – Range between two timestamps. Possible keys are *min* and *max*, with values given as time stamps in ms.
- **external_id_prefix** (*str*) – Filter by this (case-sensitive) prefix for the external ID.
- **limit** (*int, optional*) – Maximum number of time series to return. Defaults to 25. Set to -1, float("inf") or None to return all items.
- **partitions** (*int*) – Retrieve time series in parallel using this number of workers. Also requires *limit=None* to be passed.
- **include_metadata** (*bool, optional*) – Ignored. Only present in parameter list for backward compatibility.

Returns The requested time series.

Return type *TimeSeriesList*

Examples

List time series:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.time_series.list(limit=5)
```

Iterate over time series:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for ts in c.time_series:
...     ts # do something with the time_series
```

Iterate over chunks of time series to reduce memory load:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for ts_list in c.time_series(chunk_size=2500):
...     ts_list # do something with the time_series
```

Aggregate time series

TimeSeriesAPI.**aggregate**(*filter: Union[cognite.client.data_classes.time_series.TimeSeriesFilter, Dict[KT, VT]] = None*) → List[cognite.client.data_classes.time_series.TimeSeriesAggregate]

Aggregate time series

Parameters **filter** (*Union[TimeSeriesFilter, Dict]*) – Filter on time series filter with exact match

Returns List of sequence aggregates

Return type List[*TimeSeriesAggregate*]

Examples

List time series:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.time_series.aggregate(filter={"unit": "kpa"})
```

Search for time series

TimeSeriesAPI.**search**(*name: str = None, description: str = None, query: str = None, filter: Union[cognite.client.data_classes.time_series.TimeSeriesFilter, Dict[KT, VT]] = None, limit: int = 100*) → cognite.client.data_classes.time_series.TimeSeriesList

Search for time series. Primarily meant for human-centric use-cases and data exploration, not for programs, since matching and ordering may change over time. Use the *list* function if stable or exact matches are required.

Parameters

- **name** (*str, optional*) – Prefix and fuzzy search on name.
- **description** (*str, optional*) – Prefix and fuzzy search on description.
- **query** (*str, optional*) – Search on name and description using wildcard search on each of the words (separated by spaces). Retrieves results where at least one word must match. Example: 'some other'
- **filter** (*Union[TimeSeriesFilter, Dict], optional*) – Filter to apply. Performs exact match on these fields.
- **limit** (*int, optional*) – Max number of results to return.

Returns List of requested time series.

Return type *TimeSeriesList*

Examples

Search for a time series:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.time_series.search(name="some name")
```

Search for all time series connected to asset with id 123:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.time_series.search(filter={"asset_ids":[123]})
```

Create time series

TimeSeriesAPI.**create**(*time_series: Union[cognite.client.data_classes.time_series.TimeSeries, List[cognite.client.data_classes.time_series.TimeSeries]]*) → Union[cognite.client.data_classes.time_series.TimeSeries, cognite.client.data_classes.time_series.TimeSeriesList]

Create one or more time series.

Parameters **time_series** (*Union[TimeSeries, List[TimeSeries]]*) – TimeSeries or list of TimeSeries to create.

Returns The created time series.

Return type Union[*TimeSeries*, *TimeSeriesList*]

Examples

Create a new time series:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TimeSeries
>>> c = CogniteClient()
>>> ts = c.time_series.create(TimeSeries(name="my ts"))
```

Delete time series

TimeSeriesAPI.**delete**(*id: Union[int, List[int]] = None*, *external_id: Union[str, List[str]] = None*, *ignore_unknown_ids: bool = False*) → None

Delete one or more time series.

Parameters

- **id** (*Union[int, List[int]]*) – Id or list of ids
- **external_id** (*Union[str, List[str]]*) – External ID or list of external ids
- **ignore_unknown_ids** (*bool*) – Ignore IDs and external IDs that are not found rather than throw an exception.

Returns None

Examples

Delete time series by id or external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.time_series.delete(id=[1,2,3], external_id="3")
```

Update time series

TimeSeriesAPI.**update**(*item: Union[cognite.client.data_classes.time_series.TimeSeries, cognite.client.data_classes.time_series.TimeSeriesUpdate, List[Union[cognite.client.data_classes.time_series.TimeSeries, cognite.client.data_classes.time_series.TimeSeriesUpdate]]]*) → Union[cognite.client.data_classes.time_series.TimeSeries, cognite.client.data_classes.time_series.TimeSeriesList]

Update one or more time series.

Parameters **item** (*Union[TimeSeries, TimeSeriesUpdate, List[Union[TimeSeries, TimeSeriesUpdate]]]*) – Time series to update

Returns Updated time series.

Return type Union[*TimeSeries*, *TimeSeriesList*]

Examples

Update a time series that you have fetched. This will perform a full update of the time series:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.time_series.retrieve(id=1)
>>> res.description = "New description"
>>> res = c.time_series.update(res)
```

Perform a partial update on a time series, updating the description and adding a new field to metadata:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TimeSeriesUpdate
>>> c = CogniteClient()
>>> my_update = TimeSeriesUpdate(id=1).description.set("New description").
↪metadata.add({"key": "value"})
>>> res = c.time_series.update(my_update)
```

Data classes

class cognite.client.data_classes.time_series.**TimeSeries**(*id: int = None, external_id: str = None, name: str = None, is_string: bool = None, metadata: Dict[str, str] = None, unit: str = None, asset_id: int = None, is_step: bool = None, description: str = None, security_categories: List[int] = None, data_set_id: int = None, created_time: int = None, last_updated_time: int = None, legacy_name: str = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

No description.

Parameters

- **id** (*int*) – A server-generated ID for the object.
- **external_id** (*str*) – The externally supplied ID for the time series.
- **name** (*str*) – The display short name of the time series. Note: Value of this field can differ from name presented by older versions of API 0.3-0.6.
- **is_string** (*bool*) – Whether the time series is string valued or not.
- **metadata** (*Dict[str, str]*) – Custom, application specific metadata. String key -> String value. Limits: Maximum length of key is 32 bytes, value 512 bytes, up to 16 key-value pairs.
- **unit** (*str*) – The physical unit of the time series.
- **asset_id** (*int*) – Asset ID of equipment linked to this time series.
- **is_step** (*bool*) – Whether the time series is a step series or not.
- **description** (*str*) – Description of the time series.
- **security_categories** (*List[int]*) – The required security categories to access this time series.
- **data_set_id** (*int*) – The dataSet Id for the item.
- **created_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.

- **last_updated_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **legacy_name** (*str*) – Set a value for legacyName to allow applications using API v0.3, v04, v05, and v0.6 to access this time series. The legacy name is the human-readable name for the time series and is mapped to the name field used in API versions 0.3-0.6. The legacyName field value must be unique, and setting this value to an already existing value will return an error. We recommend that you set this field to the same value as externalId.
- **cognite_client** (CogniteClient) – The client to associate with this object.

asset() → Asset
Returns the asset this time series belongs to.

Returns The asset given by its *asset_id*.

Return type *Asset*

count() → int
Returns the number of datapoints in this time series.

This result may not be completely accurate, as it is based on aggregates which may be occasionally out of date.

Returns The number of datapoints in this time series.

Return type int

first() → Optional[Datapoint]
Returns the first datapoint in this time series.

Returns A datapoint object containing the value and timestamp of the first datapoint.

Return type *Datapoint*

latest() → Optional[Datapoint]
Returns the latest datapoint in this time series

Returns A datapoint object containing the value and timestamp of the latest datapoint.

Return type *Datapoint*

class cognite.client.data_classes.time_series.**TimeSeriesAggregate**(*count: int = None*, ***kwargs*)

Bases: dict

No description.

Parameters **count** (*int*) – No description.

class cognite.client.data_classes.time_series.**TimeSeriesFilter**(*name: str = None, unit: str = None, is_string: bool = None, is_step: bool = None, metadata: Dict[str, str] = None, asset_ids: List[int] = None, asset_external_ids: List[str] = None, root_asset_ids: List[int] = None, asset_subtree_ids: List[Dict[str, Any]] = None, data_set_ids: List[Dict[str, Any]] = None, external_id_prefix: str = None, created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.Timestam = None, last_updated_time: Union[Dict[str, Any], cognite.client.data_classes.shared.Timestam = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteFilter*

No description.

Parameters

- **name** (*str*) – Filter on name.
- **unit** (*str*) – Filter on unit.
- **is_string** (*bool*) – Filter on isString.
- **is_step** (*bool*) – Filter on isStep.
- **metadata** (*Dict[str, str]*) – Custom, application specific metadata. String key -> String value. Limits: Maximum length of key is 32 bytes, value 512 bytes, up to 16 key-value pairs.
- **asset_ids** (*List[int]*) – Only include time series that reference these specific asset IDs.

- **asset_external_ids** (*List[str]*) – Asset External IDs of related equipment that this time series relates to.
- **root_asset_ids** (*List[int]*) – Only include time series that have a related asset in a tree rooted at any of these root assetIds.
- **asset_subtree_ids** (*List[Dict[str, Any]]*) – Only include time series that are related to an asset in a subtree rooted at any of these assetIds (including the roots given). If the total size of the given subtrees exceeds 100,000 assets, an error will be returned.
- **data_set_ids** (*List[Dict[str, Any]]*) – No description.
- **external_id_prefix** (*str*) – Filter by this (case-sensitive) prefix for the external ID.
- **created_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **last_updated_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.time_series.**TimeSeriesList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.time_series.**TimeSeriesUpdate**(*id: int = None, external_id: str = None*)

Bases: *cognite.client.data_classes._base.CogniteUpdate*

Changes will be applied to time series.

Parameters

- **id** (*int*) – A server-generated ID for the object.
- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.

2.4.8 Data points

Retrieve datapoints

DatapointsAPI.**retrieve**(*start: Union[int, str, datetime.datetime], end: Union[int, str, datetime.datetime], id: Union[int, List[int], Dict[str, Union[int, List[str]]], List[Dict[str, Union[int, List[str]]]]] = None, external_id: Union[str, List[str], Dict[str, Union[int, List[str]]], List[Dict[str, Union[int, List[str]]]]] = None, aggregates: List[str] = None, granularity: str = None, include_outside_points: bool = None, limit: int = None, ignore_unknown_ids: bool = False*) → Union[None, cognite.client.data_classes.datapoints.Datapoints, cognite.client.data_classes.datapoints.DatapointsList]

Get datapoints for one or more time series.

Note that you cannot specify the same ids/external_ids multiple times.

Parameters

- **start** (*Union[int, str, datetime]*) – Inclusive start.

- **end** (*Union[int, str, datetime]*) – Exclusive end.
- **id** (*Union[int, List[int], Dict[str, Any], List[Dict[str, Any]]]*) – Id or list of ids. Can also be object specifying aggregates. See example below.
- **external_id** (*Union[str, List[str], Dict[str, Any], List[Dict[str, Any]]]*) – External id or list of external ids. Can also be object specifying aggregates. See example below.
- **aggregates** (*List[str]*) – List of aggregate functions to apply.
- **granularity** (*str*) – The granularity to fetch aggregates at. e.g. '1s', '2h', '10d'.
- **include_outside_points** (*bool*) – Whether or not to include outside points.
- **limit** (*int*) – Maximum number of datapoints to return for each time series.
- **ignore_unknown_ids** (*bool*) – Ignore IDs and external IDs that are not found rather than throw an exception.

Returns A Datapoints object containing the requested data, or a list of such objects. If *ignore_unknown_id* is True, single id is requested and it is not found, the function will return *None*.

Return type Union[None, *Datapoints*, *DatapointsList*]

Examples

You can get specify the ids of the datapoints you wish to retrieve in a number of ways. In this example we are using the time-ago format to get raw data for the time series with id 1:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> dps = c.datapoints.retrieve(id=1, start="2w-ago", end="now")
```

We can also get aggregated values, such as average. Here we are getting daily averages for all of 2018 for two different time series. Note that we are fetching them using their external ids:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> dps = c.datapoints.retrieve(external_id=["abc", "def"],
...                             start=datetime(2018,1,1),
...                             end=datetime(2019,1,1),
...                             aggregates=["average"],
...                             granularity="1d")
```

If you want different aggregates for different time series specify your ids like this:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> dps = c.datapoints.retrieve(id=[{"id": 1, "aggregates": ["average"]},
...                                 {"id": 1, "aggregates": ["min"]}],
...                             external_id={"externalId": "1", "aggregates": ["max"]}
↪,
...                             start="1d-ago", end="now", granularity="1h")
```

Retrieve pandas dataframe

DatapointsAPI.**retrieve_dataframe**(*start: Union[int, str, datetime.datetime], end: Union[int, str, datetime.datetime], aggregates: List[str], granularity: str, id: Union[int, List[int], Dict[str, Union[int, List[str]]], List[Dict[str, Union[int, List[str]]]]] = None, external_id: Union[str, List[str], Dict[str, Union[int, List[str]]], List[Dict[str, Union[int, List[str]]]]] = None, limit: int = None, include_aggregate_name=True, complete: str = None, ignore_unknown_ids: bool = False*) → pandas.DataFrame

Get a pandas dataframe describing the requested data.

Note that you cannot specify the same ids/external_ids multiple times.

Parameters

- **start** (*Union[int, str, datetime]*) – Inclusive start.
- **end** (*Union[int, str, datetime]*) – Exclusive end.
- **aggregates** (*List[str]*) – List of aggregate functions to apply.
- **granularity** (*str*) – The granularity to fetch aggregates at. e.g. '1s', '2h', '10d'.
- **id** (*Union[int, List[int], Dict[str, Any], List[Dict[str, Any]]]*) – Id or list of ids. Can also be object specifying aggregates. See example below.
- **external_id** (*Union[str, List[str], Dict[str, Any], List[Dict[str, Any]]]*) – External id or list of external ids. Can also be object specifying aggregates. See example below.
- **limit** (*int*) – Maximum number of datapoints to return for each time series.
- **include_aggregate_name** (*bool*) – Include 'aggregate' in the column name. Defaults to True and should only be set to False when only a single aggregate is requested per id/external id.
- **complete** (*str*) – Post-processing of the dataframe.
- **ignore_unknown_ids** (*bool*) – Ignore IDs and external IDs that are not found rather than throw an exception.

  Pass 'fill' to insert missing entries into the index, and complete data where possible (supports interpolation, stepInterpolation, count, sum, totalVariation).

  Pass 'fill,dropna' to additionally drop rows in which any aggregate for any time series has missing values (typically rows at the start and end for interpolation aggregates). This option guarantees that all returned dataframes have the exact same shape and no missing values anywhere, and is only supported for aggregates sum, count, totalVariation, interpolation and stepInterpolation.

Returns The requested dataframe

Return type pandas.DataFrame

Examples

Get a pandas dataframe:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> df = c.datapoints.retrieve_dataframe(id=[1,2,3], start="2w-ago", end="now",
...         aggregates=["average","sum"], granularity="1h")
```

Get a pandas dataframe with the index regularly spaced at 1 minute intervals, missing values completed and without the aggregate name in the columns:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> df = c.datapoints.retrieve_dataframe(id=[1,2,3], start="2w-ago", end="now",
...         aggregates=["interpolation"], granularity="1m", include_aggregate_
↪name=False, complete="fill,dropna")
```

Retrieve pandas dataframes indexed by aggregate

DatapointsAPI.**retrieve_dataframe_dict**(*start: Union[int, str, datetime.datetime], end: Union[int, str, datetime.datetime], aggregates: List[str], granularity: str, id: Union[int, List[int], Dict[str, Union[int, List[str]]], List[Dict[str, Union[int, List[str]]]]] = None, external_id: Union[str, List[str], Dict[str, Union[int, List[str]]], List[Dict[str, Union[int, List[str]]]]] = None, limit: int = None, ignore_unknown_ids: bool = False, complete: bool = None*) → Dict[str, pandas.DataFrame]

Get a dictionary of aggregate: pandas dataframe describing the requested data.

Parameters

- **start** (*Union[int, str, datetime]*) – Inclusive start.
- **end** (*Union[int, str, datetime]*) – Exclusive end.
- **aggregates** (*List[str]*) – List of aggregate functions to apply.
- **granularity** (*str*) – The granularity to fetch aggregates at. e.g. '1s', '2h', '10d'.
- **(Union[int, List[int], Dict[str, Any], List[Dict[str, Any]]]** (*id*) – Id or list of ids. Can also be object specifying aggregates.
- **external_id** (*Union[str, List[str], Dict[str, Any], List[Dict[str, Any]]]*) – External id or list of external ids. Can also be object specifying aggregates.
- **limit** (*int*) – Maximum number of datapoints to return for each time series.
- **ignore_unknown_ids** (*bool*) – Ignore IDs and external IDs that are not found rather than throw an exception.
- **complete** (*str*) – Post-processing of the dataframe.

  Pass 'fill' to insert missing entries into the index, and complete data where possible (supports interpolation, stepInterpolation, count, sum, totalVariation).

  Pass 'fill,dropna' to additionally drop rows in which any aggregate for any time series has missing values (typically rows at the start and end for interpolation aggregates). This option guarantees that all returned dataframes have the exact same shape and no missing values anywhere, and is only supported for aggregates sum, count, totalVariation, interpolation and stepInterpolation.

Returns A dictionary of aggregate: dataframe.

Return type Dict[str,pandas.DataFrame]

Examples

Get a dictionary of pandas dataframes, with the index evenly spaced at 1h intervals, missing values completed in the middle and incomplete entries dropped at the start and end:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> dfs = c.datapoints.retrieve_dataframe_dict(id=[1,2,3], start="2w-ago", end=
↪"now",
...           aggregates=["interpolation","count"], granularity="1h", complete=
↪"fill,dropna")
```

Perform data points queries

`DatapointsAPI.`**`query`**(*query: Union[cognite.client.data_classes.datapoints.DatapointsQuery, List[cognite.client.data_classes.datapoints.DatapointsQuery]]*) → Union[cognite.client.data_classes.datapoints.DatapointsList, List[cognite.client.data_classes.datapoints.DatapointsList]]

Get datapoints for one or more time series

This method is different from get() in that you can specify different start times, end times, and granularities for each requested time series.

Parameters **`query`** (*Union[DatapointsQuery, List[DatapointsQuery]]*) – List of datapoint queries.

Returns The requested DatapointsList(s).

Return type Union[*DatapointsList*, List[*DatapointsList*]]

Examples

This method is useful if you want to get multiple time series, but you want to specify different starts, ends, or granularities for each. e.g.:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import DatapointsQuery
>>> c = CogniteClient()
>>> queries = [DatapointsQuery(id=1, start="2d-ago", end="now"),
...            DatapointsQuery(external_id="abc",
...                            start="10d-ago",
...                            end="now",
...                            aggregates=["average"],
...                            granularity="1m")]
>>> res = c.datapoints.query(queries)
```

Retrieve latest datapoint

DatapointsAPI.**retrieve_latest**(*id: Union[int, List[int]] = None, external_id: Union[str, List[str]] = None, before: Union[int, str, datetime.datetime] = None, ignore_unknown_ids: bool = False*) → Union[cognite.client.data_classes.datapoints.Datapoints, cognite.client.data_classes.datapoints.DatapointsList]

Get the latest datapoint for one or more time series

Parameters

- **(Union[int, List[int]]** (*id*) – Id or list of ids.
- **external_id** (*Union[str, List[str]*) – External id or list of external ids.
- **before** – Union[int, str, datetime]: Get latest datapoint before this time.
- **ignore_unknown_ids** (*bool*) – Ignore IDs and external IDs that are not found rather than throw an exception.

Returns A Datapoints object containing the requested data, or a list of such objects.

Return type Union[*Datapoints*, *DatapointsList*]

Examples

Getting the latest datapoint in a time series. This method returns a Datapoints object, so the datapoint will be the first element:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.datapoints.retrieve_latest(id=1)[0]
```

You can also get the first datapoint before a specific time:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.datapoints.retrieve_latest(id=1, before="2d-ago")[0]
```

If you need the latest datapoint for multiple time series simply give a list of ids. Note that we are using external ids here, but either will work:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.datapoints.retrieve_latest(external_id=["abc", "def"])
>>> latest_abc = res[0][0]
>>> latest_def = res[1][0]
```

Insert data points

DatapointsAPI.**insert**(*datapoints: Union[List[Dict[Union[int, float, datetime.datetime], Union[int, float, str]]], List[Tuple[Union[int, float, datetime.datetime], Union[int, float, str]]]], id: int = None, external_id: str = None*) → None

Insert datapoints into a time series

Timestamps can be represented as milliseconds since epoch or datetime objects.

Parameters

- **datapoints** (*Union[List[Dict], List[Tuple],Datapoints]*) – The datapoints you wish to insert. Can either be a list of tuples, a list of dictionaries, or a Datapoints object. See examples below.
- **id** (*int*) – Id of time series to insert datapoints into.
- **external_id** (*str*) – External id of time series to insert datapoint into.

Returns None

Examples

Your datapoints can be a list of tuples where the first element is the timestamp and the second element is the value:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> # with datetime objects
>>> datapoints = [(datetime(2018,1,1), 1000), (datetime(2018,1,2), 2000)]
>>> c.datapoints.insert(datapoints, id=1)
>>> # with ms since epoch
>>> datapoints = [(150000000000, 1000), (160000000000, 2000)]
>>> c.datapoints.insert(datapoints, id=2)
```

Or they can be a list of dictionaries:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> # with datetime objects
>>> datapoints = [{"timestamp": datetime(2018,1,1), "value": 1000},
...     {"timestamp": datetime(2018,1,2), "value": 2000}]
>>> c.datapoints.insert(datapoints, external_id="abc")
>>> # with ms since epoch
>>> datapoints = [{"timestamp": 150000000000, "value": 1000},
...     {"timestamp": 160000000000, "value": 2000}]
>>> c.datapoints.insert(datapoints, external_id="def")
```

Or they can be a Datapoints object:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> data = c.datapoints.retrieve(external_id="abc",start=datetime(2018,1,1),
↪end=datetime(2018,2,2))
>>> c.datapoints.insert(data, external_id="def")
```

Insert data points into multiple time series

`DatapointsAPI.`**`insert_multiple`**(*datapoints: List[Dict[str, Union[str, int, List[T]]]]*) → None

Insert datapoints into multiple time series

Parameters **datapoints** (*List[Dict]*) – The datapoints you wish to insert along with the ids of the time series. See examples below.

Returns None

Examples

Your datapoints can be a list of tuples where the first element is the timestamp and the second element is the value:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()

>>> datapoints = []
>>> # with datetime objects and id
>>> datapoints.append({"id": 1, "datapoints": [(datetime(2018,1,1), 1000),␣
↪(datetime(2018,1,2), 2000)]})
>>> # with ms since epoch and externalId
>>> datapoints.append({"externalId": 1, "datapoints": [(150000000000, 1000),␣
↪(160000000000, 2000)]})

>>> c.datapoints.insert_multiple(datapoints)
```

Or they can be a list of dictionaries:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()

>>> datapoints = []
>>> # with datetime objects and external id
>>> datapoints.append({"externalId": "1", "datapoints": [{"timestamp":␣
↪datetime(2018,1,1), "value": 1000},
...                       {"timestamp": datetime(2018,1,2), "value": 2000}]})
>>> # with ms since epoch and id
>>> datapoints.append({"id": 1, "datapoints": [{"timestamp": 150000000000, "value
↪": 1000},
...                       {"timestamp": 160000000000, "value": 2000}]})

>>> c.datapoints.insert_multiple(datapoints)
```

Insert pandas dataframe

DatapointsAPI.**insert_dataframe**(*dataframe*, *external_id_headers: bool = False*)

Insert a dataframe.

The index of the dataframe must contain the timestamps. The names of the remaining columns specify the ids or external ids of the time series to which column contents will be written.

Said time series must already exist.

Parameters

- **dataframe** (*pandas.DataFrame*) – Pandas DataFrame Object containing the time series.
- **external_id_headers** (*bool*) – Set to True if the column headers are external ids rather than internal ids. Defaults to False.

Returns None

Examples

Post a dataframe with white noise:

```
>>> import numpy as np
>>> import pandas as pd
>>> from cognite.client import CogniteClient
>>> from datetime import datetime, timedelta
>>>
>>> c = CogniteClient()
>>> ts_id = 123
>>> start = datetime(2018, 1, 1)
>>> x = pd.DatetimeIndex([start + timedelta(days=d) for d in range(100)])
>>> y = np.random.normal(0, 1, 100)
>>> df = pd.DataFrame({ts_id: y}, index=x)
>>> c.datapoints.insert_dataframe(df)
```

Delete a range of data points

DatapointsAPI.**delete_range**(*start: Union[int, str, datetime.datetime], end: Union[int, str, datetime.datetime], id: int = None, external_id: str = None*) → None

Delete a range of datapoints from a time series.

Parameters

- **start** (*Union[int, str, datetime]*) – Inclusive start of delete range
- **end** (*Union[int, str, datetime]*) – Exclusvie end of delete range
- **id** (*int*) – Id of time series to delete data from
- **external_id** (*str*) – External id of time series to delete data from

Returns None

Examples

Deleting the last week of data from a time series:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.datapoints.delete_range(start="1w-ago", end="now", id=1)
```

Delete ranges of data points

DatapointsAPI.**delete_ranges**(*ranges: List[Dict[str, Any]]*) → None

Delete a range of datapoints from multiple time series.

Parameters **ranges** (*List[Dict[str, Any]]*) – The ids an ranges to delete. See examples below.

Returns None

Examples

Each element in the list ranges must be specify either id or externalId, and a range:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> ranges = [{"id": 1, "start": "2d-ago", "end": "now"},
...           {"externalId": "abc", "start": "2d-ago", "end": "now"}]
>>> c.datapoints.delete_ranges(ranges)
```

Data classes

class cognite.client.data_classes.datapoints.**Datapoint**(*timestamp: Union[int, float] = None*, *value: Union[str, int, float] = None*, *average: float = None*, *max: float = None*, *min: float = None*, *count: int = None*, *sum: float = None*, *interpolation: float = None*, *step_interpolation: float = None*, *continuous_variance: float = None*, *discrete_variance: float = None*, *total_variation: float = None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

An object representing a datapoint.

Parameters

- **timestamp** (*Union[int, float]*) – The data timestamp in milliseconds since the epoch (Jan 1, 1970).
- **value** (*Union[str, int, float]*) – The data value. Can be String or numeric depending on the metric
- **average** (*float*) – The integral average value in the aggregate period
- **max** (*float*) – The maximum value in the aggregate period
- **min** (*float*) – The minimum value in the aggregate period
- **count** (*int*) – The number of datapoints in the aggregate period
- **sum** (*float*) – The sum of the datapoints in the aggregate period
- **interpolation** (*float*) – The interpolated value of the series in the beginning of the aggregate
- **step_interpolation** (*float*) – The last value before or at the beginning of the aggregate.
- **continuous_variance** (*float*) – The variance of the interpolated underlying function.
- **discrete_variance** (*float*) – The variance of the datapoint values.
- **total_variation** (*float*) – The total variation of the interpolated underlying function.

to_pandas(*camel_case=True*) → pandas.DataFrame

Convert the datapoint into a pandas DataFrame. camel_case (bool): Convert column names to camel case (e.g. *stepInterpolation* instead of *step_interpolation*)

Returns The dataframe.

Return type pandas.DataFrame

class cognite.client.data_classes.datapoints.**Datapoints**(*id: int = None*, *external_id: str = None*, *is_string: bool = None*, *is_step: bool = None*, *unit: str = None*, *timestamp: List[Union[int, float]] = None*, *value: List[Union[int, str, float]] = None*, *average: List[float] = None*, *max: List[float] = None*, *min: List[float] = None*, *count: List[int] = None*, *sum: List[float] = None*, *interpolation: List[float] = None*, *step_interpolation: List[float] = None*, *continuous_variance: List[float] = None*, *discrete_variance: List[float] = None*, *total_variation: List[float] = None*, *error: List[Union[None, str]] = None*)

Bases: object

An object representing a list of datapoints.

Parameters

- **id** (*int*) – Id of the timeseries the datapoints belong to
- **external_id** (*str*) – External id of the timeseries the datapoints belong to (Only if id is not set)
- **is_string** (*bool*) – Whether the time series is string valued or not.
- **is_step** (*bool*) – Whether the time series is a step series or not.
- **unit** (*str*) – The physical unit of the time series.
- **timestamp** (*List[Union[int, float]]*) – The data timestamps in milliseconds since the epoch (Jan 1, 1970).
- **value** (*List[Union[int, str, float]]*) – The data values. Can be String or numeric depending on the metric
- **average** (*List[float]*) – The integral average values in the aggregate period
- **max** (*List[float]*) – The maximum values in the aggregate period
- **min** (*List[float]*) – The minimum values in the aggregate period
- **count** (*List[int]*) – The number of datapoints in the aggregate periods

- **sum** (*List[float]*) – The sum of the datapoints in the aggregate periods
- **interpolation** (*List[float]*) – The interpolated values of the series in the beginning of the aggregates
- **step_interpolation** (*List[float]*) – The last values before or at the beginning of the aggregates.
- **continuous_variance** (*List[float]*) – The variance of the interpolated underlying function.
- **discrete_variance** (*List[float]*) – The variance of the datapoint values.
- **total_variation** (*List[float]*) – The total variation of the interpolated underlying function.

dump(*camel_case: bool = False*) → Dict[str, Any]
Dump the datapoints into a json serializable Python data type.

Parameters **camel_case** (*bool*) – Use camelCase for attribute names. Defaults to False.

Returns A list of dicts representing the instance.

Return type List[Dict[str, Any]]

plot(**args*, ***kwargs*) → None
Plot the datapoints.

to_pandas(*column_names: str = 'externalId'*, *include_aggregate_name: bool = True*, *include_errors: bool = False*) → pandas.DataFrame
Convert the datapoints into a pandas DataFrame.

Parameters

- **column_names** (*str*) – Which field to use as column header. Defaults to "externalId", can also be "id".
- **include_aggregate_name** (*bool*) – Include aggregate in the column name
- **include_errors** (*bool*) – For synthetic datapoint queries, include a column with errors.

Returns The dataframe.

Return type pandas.DataFrame

class cognite.client.data_classes.datapoints.**DatapointsList**(*resources: List[Any]*, *cognite_client=None*)
Bases: *cognite.client.data_classes._base.CogniteResourceList*

plot(**args*, ***kwargs*) → None
Plot the list of datapoints.

to_pandas(*column_names: str = 'externalId'*, *include_aggregate_name: bool = True*) → pandas.DataFrame
Convert the datapoints list into a pandas DataFrame.

Parameters

- **column_names** (*str*) – Which field to use as column header. Defaults to "externalId", can also be "id".
- **include_aggregate_name** (*bool*) – Include aggregate in the column name

Returns The datapoints list as a pandas DataFrame.

Return type pandas.DataFrame

class cognite.client.data_classes.datapoints.**DatapointsQuery**(*start: Union[str, int, datetime.datetime], end: Union[str, int, datetime.datetime], id: Union[int, List[int], Dict[str, Union[int, List[str]]], List[Dict[str, Union[int, List[str]]]]] = None, external_id: Union[str, List[str], Dict[str, Union[int, List[str]]], List[Dict[str, Union[int, List[str]]]]] = None, limit: int = None, aggregates: List[str] = None, granularity: str = None, include_outside_points: bool = None, ignore_unknown_ids: bool = False*)

Bases: *cognite.client.data_classes._base.CogniteResource*

Parameters describing a query for datapoints.

Parameters

- **start** (*Union[str, int, datetime]*) – Get datapoints after this time. Format is N[timeunit]-ago where timeunit is w,d,h,m,s. Example: '2d-ago' will get everything that is up to 2 days old. Can also send time in ms since epoch.
- **end** (*Union[str, int, datetime]*) – Get datapoints up to this time. The format is the same as for start.
- **(Union[int, List[int], Dict[str, Any], List[Dict[str, Any]]]** (*id*) –

  Id or list of ids. Can also be object specifying aggregates. See example below.

  external_id (Union[str, List[str], Dict[str, Any], List[Dict[str, Any]]]): External id or list of external ids. Can also be object specifying aggregates. See example below.
- **limit** (*int*) – Return up to this number of datapoints.
- **aggregates** (*List[str]*) – The aggregates to be returned. Use default if null. An empty string must be sent to get raw data if the default is a set of aggregates.
- **granularity** (*str*) – The granularity size and granularity of the aggregates.
- **include_outside_points** (*bool*) – Whether to include the last datapoint before the requested time period,and the first one after the requested period. This can be useful for

interpolating data. Not available for aggregates.

- **ignore_unknown_ids** (*bool*) – Ignore IDs and external IDs that are not found rather than throw an exception. Note that in this case the function always returns a DatapointsList even when a single id is requested.

2.4.9 Sequences

Retrieve a sequence by id

SequencesAPI.**retrieve**(*id: Optional[int] = None*, *external_id: Optional[str] = None*) → Optional[cognite.client.data_classes.sequences.Sequence]

Retrieve a single sequence by id.

Parameters

- **id** (*int, optional*) – ID
- **external_id** (*str, optional*) – External ID

Returns Requested sequences or None if it does not exist.

Return type Optional[*Sequence*]

Examples

Get sequences by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.sequences.retrieve(id=1)
```

Get sequences by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.sequences.retrieve(external_id="1")
```

Retrieve multiple sequences by id

SequencesAPI.**retrieve_multiple**(*ids: Optional[List[int]] = None*, *external_ids: Optional[List[str]] = None*) → cognite.client.data_classes.sequences.SequenceList

Retrieve multiple sequences by id.

Parameters

- **ids** (*List[int], optional*) – IDs
- **external_ids** (*List[str], optional*) – External IDs

Returns The requested sequences.

Return type *SequenceList*

Examples

Get sequences by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.sequences.retrieve_multiple(ids=[1, 2, 3])
```

Get sequences by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.sequences.retrieve_multiple(external_ids=["abc", "def"])
```

List sequences

SequencesAPI.**list**(*name: str = None, external_id_prefix: str = None, metadata: Dict[str, str] = None, asset_ids: List[int] = None, root_asset_ids: List[int] = None, asset_subtree_ids: List[int] = None, asset_subtree_external_ids: List[str] = None, data_set_ids: List[int] = None, data_set_external_ids: List[str] = None, created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, last_updated_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange] = None, limit: Optional[int] = 25*) → cognite.client.data_classes.sequences.SequenceList

Iterate over sequences

Fetches sequences as they are iterated over, so you keep a limited number of objects in memory.

Parameters

- **name** (*str*) – Filter out sequences that do not have this *exact* name.
- **external_id_prefix** (*str*) – Filter out sequences that do not have this string as the start of the externalId
- **metadata** (*Dict[str, Any]*) – Filter out sequences that do not match these metadata fields and values (case-sensitive). Format is {"key1":"value1","key2":"value2"}.
- **asset_ids** (*List[int]*) – Filter out sequences that are not linked to any of these assets.
- **root_asset_ids** (*List[int]*) – Filter out sequences not linked to assets with one of these assets as the root asset.
- **asset_subtree_ids** (*List[int]*) – List of asset subtrees ids to filter on.
- **asset_subtree_external_ids** (*List[str]*) – List of asset subtrees external ids to filter on.
- **data_set_ids** (*List[int]*) – Return only events in the specified data sets with these ids.
- **data_set_external_ids** (*List[str]*) – Return only events in the specified data sets with these external ids.
- **created_time** (*Union[Dict[str, int], TimestampRange]*) – Range between two timestamps. Possible keys are *min* and *max*, with values given as time stamps in ms.

- **last_updated_time** (*Union[Dict[str, int], TimestampRange]*) – Range between two timestamps. Possible keys are *min* and *max*, with values given as time stamps in ms.
- **limit** (*int, optional*) – Max number of sequences to return. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns The requested sequences.

Return type *SequenceList*

Examples

List sequences:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.sequences.list(limit=5)
```

Iterate over sequences:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for seq in c.sequences:
...     seq # do something with the sequences
```

Iterate over chunks of sequences to reduce memory load:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for seq_list in c.sequences(chunk_size=2500):
...     seq_list # do something with the sequences
```

Aggregate sequences

SequencesAPI.**aggregate**(*filter: Union[cognite.client.data_classes.sequences.SequenceFilter, Dict[KT, VT]] = None*) → List[cognite.client.data_classes.sequences.SequenceAggregate]

Aggregate sequences

Parameters **filter** (*Union[SequenceFilter, Dict]*) – Filter on sequence filter with exact match

Returns List of sequence aggregates

Return type List[*SequenceAggregate*]

Examples

Aggregate sequences:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.sequences.aggregate(filter={"external_id_prefix": "prefix"})
```

Search for sequences

SequencesAPI.**search**(*name: str = None*, *description: str = None*, *query: str = None*, *filter: Union[cognite.client.data_classes.sequences.SequenceFilter, Dict[KT, VT]] = None*, *limit: int = 100*) → cognite.client.data_classes.sequences.SequenceList

Search for sequences. Primarily meant for human-centric use-cases and data exploration, not for programs, since matching and ordering may change over time. Use the *list* function if stable or exact matches are required.

Parameters

- **name** (*str, optional*) – Prefix and fuzzy search on name.
- **description** (*str, optional*) – Prefix and fuzzy search on description.
- **query** (*str, optional*) – Search on name and description using wildcard search on each of the words (separated by spaces). Retrieves results where at least one word must match. Example: 'some other'
- **filter** (*Union[SequenceFilter, Dict], optional*) – Filter to apply. Performs exact match on these fields.
- **limit** (*int, optional*) – Max number of results to return.

Returns List of requested sequences.

Return type *SequenceList*

Examples

Search for a sequence:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.sequences.search(name="some name")
```

Create a sequence

SequencesAPI.**create**(*sequence: Union[cognite.client.data_classes.sequences.Sequence, List[cognite.client.data_classes.sequences.Sequence]]*) → Union[cognite.client.data_classes.sequences.Sequence, cognite.client.data_classes.sequences.SequenceList]

Create one or more sequences.

Parameters **sequence** (*Union[Sequence, List[Sequence]]*) – Sequence or list of Sequence to create. The Sequence columns parameter is a list of objects with fields *externalId* (external id of the column, when omitted, they will be given ids of 'column0, column1, ...'), *valueType* (data type of the column, either STRING, LONG, or DOUBLE, with default DOUBLE), *name*, *description*, *metadata* (optional fields to describe and store information about the data in the column). Other fields will be removed automatically, so a columns definition from a different sequence object can be passed here.

Returns The created sequences.

Return type Union[*Sequence*, *SequenceList*]

Examples

Create a new sequence:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import Sequence
>>> c = CogniteClient()
>>> column_def = [{"valueType":"STRING","externalId":"user","description":"some␣
↪description"}, {"valueType":"DOUBLE","externalId":"amount"}]
>>> seq = c.sequences.create(Sequence(external_id="my_sequence", columns=column_
↪def))
```

Create a new sequence with the same column specifications as an existing sequence:

```
>>> seq2 = c.sequences.create(Sequence(external_id="my_copied_sequence",␣
↪columns=seq.columns))
```

Delete sequences

SequencesAPI.**delete**(*id: Union[int, List[int]] = None*, *external_id: Union[str, List[str]] = None*) → None

Delete one or more sequences.

Parameters

- **id** (*Union[int, List[int]]*) – Id or list of ids
- **external_id** (*Union[str, List[str]]*) – External ID or list of external ids

Returns None

Examples

Delete sequences by id or external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.sequences.delete(id=[1,2,3], external_id="3")
```

Update sequences

SequencesAPI.**update**(*item: Union[cognite.client.data_classes.sequences.Sequence, cognite.client.data_classes.sequences.SequenceUpdate, List[Union[cognite.client.data_classes.sequences.Sequence, cognite.client.data_classes.sequences.SequenceUpdate]]]*) → Union[cognite.client.data_classes.sequences.Sequence, cognite.client.data_classes.sequences.SequenceList]

Update one or more sequences.

Parameters **item** (*Union[Sequence, SequenceUpdate, List[Union[Sequence, SequenceUpdate]]]*) – Sequences to update

Returns Updated sequences.

Return type Union[*Sequence*, *SequenceList*]

Examples

Update a sequence that you have fetched. This will perform a full update of the sequences:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.sequences.retrieve(id=1)
>>> res.description = "New description"
>>> res = c.sequences.update(res)
```

Perform a partial update on a sequence, updating the description and adding a new field to metadata:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import SequenceUpdate
>>> c = CogniteClient()
>>> my_update = SequenceUpdate(id=1).description.set("New description").metadata.
↪add({"key": "value"})
>>> res = c.sequences.update(my_update)
```

Retrieve data

SequencesDataAPI.**retrieve**(*start: int, end: Optional[int], column_external_ids: Optional[List[str]] = None, external_id: Union[str, List[str]] = None, id: Union[int, List[int]] = None, limit: int = None*) → Union[cognite.client.data_classes.sequences.SequenceData, cognite.client.data_classes.sequences.SequenceDataList]

Retrieve data from a sequence

Parameters

- **start** (*int*) – Row number to start from (inclusive).
- **end** (*Union[int, None]*) – Upper limit on the row number (exclusive). Set to None or -1 to get all rows until end of sequence.
- **column_external_ids** (*Optional[List[str]]*) – List of external id for the columns of the sequence. If 'None' is passed, all columns will be retrieved.
- **id** (*int*) – Id of sequence.
- **external_id** (*str*) – External id of sequence.
- **limit** (*int*) – Maximum number of rows to return per sequence.

Returns List of sequence data

Examples

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.sequences.data.retrieve(id=0, start=0, end=None)
>>> tuples = [(r,v) for r,v in res.items()] # You can use this iterator in for␣
↪loops and list comprehensions,
>>> single_value = res[23] # ... get the values at a single row number,
>>> col = res.get_column(external_id='columnExtId') # ... get the array of values␣
↪for a specific column,
>>> df = res.to_pandas() # ... or convert the result to a dataframe
```

Retrieve pandas dataframe

SequencesDataAPI.**retrieve_dataframe**(*start: int, end: Optional[int], column_external_ids: Optional[List[str]] = None, external_id: str = None, column_names: str = None, id: int = None, limit: int = None*)

Retrieve data from a sequence as a pandas dataframe

Parameters

- **start** (*int*) – (inclusive) row number to start from.
- **end** (*Union[int, None]*) – (exclusive) upper limit on the row number. Set to None or -1 to get all rows until end of sequence.
- **column_external_ids** (*Optional[List[str]]*) – List of external id for the columns of the sequence. If 'None' is passed, all columns will be retrieved.
- **id** (*int*) – Id of sequence
- **external_id** (*str*) – External id of sequence.
- **column_names** (*str*) – Which field(s) to use as column header. Can use "externalId", "id", "columnExternalId", "id|columnExternalId" or "externalId|columnExternalId". Default is "externalId|columnExternalId" for queries on more than one sequence, and "columnExternalId" for queries on a single sequence.
- **limit** (*int*) – Maximum number of rows to return per sequence.

Returns pandas.DataFrame

Examples

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> df = c.sequences.data.retrieve_dataframe(id=0, start=0, end=None)
```

Insert rows into a sequence

SequencesDataAPI.**insert**(*rows: Union[Dict[int, List[Union[int, str, float]]], List[Tuple[int, Union[int, float, str]]], List[Dict[str, Any]], cognite.client.data_classes.sequences.SequenceData], column_external_ids: Optional[List[str]], id: int = None, external_id: str = None*) → None

Insert rows into a sequence

Parameters

- **column_external_ids** (*Optional[List[str]]*) – List of external id for the columns of the sequence.
- **rows** (*Union[Dict[int, List[Union[int, float, str]]], List[Tuple[int,Union[int, float, str]]], List[Dict[str, Any]], SequenceData]*) – The rows you wish to insert. Can either be a list of tuples, a list of {"rowNumber":... ,"values": ...} objects, a dictionary of rowNumber: data, or a SequenceData object. See examples below.
- **id** (*int*) – Id of sequence to insert rows into.
- **external_id** (*str*) – External id of sequence to insert rows into.

Returns None

Examples

Your rows of data can be a list of tuples where the first element is the rownumber and the second element is the data to be inserted:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> seq = c.sequences.create(Sequence(columns=[{"valueType": "STRING", "externalId
↪":"col_a"},{"valueType": "DOUBLE", "externalId":"col_b"}]))
>>> data = [(1, ['pi',3.14]), (2, ['e',2.72]) ]
>>> c.sequences.data.insert(column_external_ids=["col_a","col_b"], rows=data,␣
↪id=1)
```

They can also be provided as a list of API-style objects with a rowNumber and values field:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> data = [{"rowNumber": 123, "values": ['str',3]}, {"rowNUmber": 456, "values":␣
↪["bar",42]} ]
>>> c.sequences.data.insert(data, id=1, column_external_ids=["col_a","col_b"]) #␣
↪implicit columns are retrieved from metadata
```

Or they can be a given as a dictionary with row number as the key, and the value is the data to be inserted at that row:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> data = {123 : ['str',3], 456 : ['bar',42] }
>>> c.sequences.data.insert(column_external_ids=['stringColumn','intColumn'],␣
↪rows=data, id=1)
```

Finally, they can be a SequenceData object retrieved from another request. In this case column_external_ids from this object are used as well.

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> data = c.sequences.data.retrieve(id=2,start=0,end=10)
>>> c.sequences.data.insert(rows=data, id=1,column_external_ids=None)
```

Insert a pandas dataframe into a sequence

SequencesDataAPI.**insert_dataframe**(*dataframe*, *external_id_headers: bool = True*, *id: int = None*, *external_id: str = None*) → None

Insert a Pandas dataframe.

The index of the dataframe must contain the row numbers. The names of the remaining columns specify the column external ids. The sequence and columns must already exist.

Parameters

- **dataframe** (*pandas.DataFrame*) – Pandas DataFrame object containing the sequence data.
- **external_id_headers** (*bool*) – Ignored parameter here for backwards compatibility. Dataframe columns should always match sequence column external ids.

- **id** (*int*) – Id of sequence to insert rows into.
- **external_id** (*str*) – External id of sequence to insert rows into.

Returns None

Examples

Multiply data in the sequence by 2:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> df = c.sequences.data.retrieve_dataframe(id=123, start=0, end=None)
>>> c.sequences.data.insert_dataframe(df*2, id=123)
```

Delete rows from a sequence

SequencesDataAPI.**delete**(*rows: List[int], id: int = None, external_id: str = None*) → None

Delete rows from a sequence

Parameters

- **rows** (*List[int]*) – List of row numbers.
- **id** (*int*) – Id of sequence to delete rows from.
- **external_id** (*str*) – External id of sequence to delete rows from.

Returns None

Examples

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.sequences.data.delete(id=0, rows=[1,2,42])
```

Delete a range of rows from a sequence

SequencesDataAPI.**delete_range**(*start: int, end: Optional[int], id: int = None, external_id: str = None*) → None

Delete a range of rows from a sequence. Note this operation is potentially slow, as retrieves each row before deleting.

Parameters

- **start** (*int*) – Row number to start from (inclusive).
- **end** (*Union[int, None]*) – Upper limit on the row number (exclusive). Set to None or -1 to delete all rows until end of sequence.
- **id** (*int*) – Id of sequence to delete rows from.
- **external_id** (*str*) – External id of sequence to delete rows from.

Returns None

Examples

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.sequences.data.delete_range(id=0, start=0, end=None)
```

Data classes

class cognite.client.data_classes.sequences.**Sequence**(*id: int = None, name: str = None, description: str = None, asset_id: int = None, external_id: str = None, metadata: Dict[str, Any] = None, columns: List[Dict[str, Any]] = None, created_time: int = None, last_updated_time: int = None, data_set_id: int = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

Information about the sequence stored in the database

Parameters

- **id** (*int*) – Unique cognite-provided identifier for the sequence
- **name** (*str*) – Name of the sequence
- **description** (*str*) – Description of the sequence
- **asset_id** (*int*) – Optional asset this sequence is associated with
- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.
- **metadata** (*Dict[str, Any]*) – Custom, application specific metadata. String key -> String value. Maximum length of key is 32 bytes, value 512 bytes, up to 16 key-value pairs.
- **columns** (*List[Dict[str, Any]]*) – List of column definitions
- **created_time** (*int*) – Time when this sequence was created in CDF in milliseconds since Jan 1, 1970.
- **last_updated_time** (*int*) – The last time this sequence was updated in CDF, in milliseconds since Jan 1, 1970.
- **data_set_id** (*int*) – Data set that this sequence belongs to
- **cognite_client** (CogniteClient) – The client to associate with this object.

column_external_ids
Retrieves list of column external ids for the sequence, for use in e.g. data retrieve or insert methods

Returns List of sequence column external ids

column_value_types
Retrieves list of column value types

Returns List of column value types

rows(*start: int, end: int*) → List[dict]
Retrieves rows from this sequence.

Returns List of sequence data.

class cognite.client.data_classes.sequences.**SequenceAggregate**(*count: int = None*, ***kwargs*)

Bases: dict

No description.

Parameters **count** (*int*) – No description.

class cognite.client.data_classes.sequences.**SequenceData**(*id: int = None*, *external_id: str = None*, *rows: List[dict] = None*, *row_numbers: List[int] = None*, *values: List[List[Union[int, str, float]]] = None*, *columns: List[dict] = None*)

Bases: object

An object representing a list of rows from a sequence.

Parameters

- **id** (*int*) – Id of the sequence the data belong to
- **external_id** (*str*) – External id of the sequence the data belong to
- **rows** (*List[dict]*) – Combined row numbers and row data object from the API. If you pass this, row_numbers/values are ignored.
- **row_numbers** (*List[int]*) – The data row numbers.
- **values** (*List[List[Union[int, str, float]]]*) – The data values, one row at a time.
- **columns** – List[dict]: The column information, in the format returned by the API.

column_external_ids
Retrieves list of column external ids for the sequence, for use in e.g. data retrieve or insert methods.

Returns List of sequence column external ids.

column_value_types
Retrieves list of column value types.

Returns List of column value types

dump(*camel_case: bool = False*) → Dict[str, Any]
Dump the sequence data into a json serializable Python data type.

Parameters **camel_case** (*bool*) – Use camelCase for attribute names. Defaults to False.

Returns A list of dicts representing the instance.

Return type List[Dict[str, Any]]

get_column(*external_id: str*) → List[Union[int, str, float]]
Get a column by external_id.

Parameters **external_id** (*str*) – External id of the column.

Returns A list of values for that column in the sequence

Return type List[Union[int, str, float]]

items() → Generator[Tuple[int, List[Union[int, str, float]]], None, None]
Returns an iterator over tuples of (row number, values).

to_pandas(*column_names: str = 'columnExternalId'*) → pandas.DataFrame
Convert the sequence data into a pandas DataFrame.

Parameters **column_names** (*str*) – Which field(s) to use as column header. Can use "externalId", "id", "columnExternalId", "id|columnExternalId" or "externalId|columnExternalId".

Returns The dataframe.

Return type pandas.DataFrame

class cognite.client.data_classes.sequences.**SequenceDataList**(*resources: List[Any], cognite_client=None*)
Bases: *cognite.client.data_classes._base.CogniteResourceList*

to_pandas(*column_names: str = 'externalId|columnExternalId'*) → pandas.DataFrame
Convert the sequence data list into a pandas DataFrame. Each column will be a sequence.

Parameters

- **column_names** (*str*) – Which field to use as column header. Can use any combination of "externalId", "columnExternalId", "id" and other characters as a template.
- **include_aggregate_name** (*bool*) – Include aggregate in the column name

Returns The sequence data list as a pandas DataFrame.

Return type pandas.DataFrame

class cognite.client.data_classes.sequences.**SequenceFilter**(*name: str = None, external_id_prefix: str = None, metadata: Dict[str, Any] = None, asset_ids: List[int] = None, root_asset_ids: List[int] = None, asset_subtree_ids: List[Dict[str, Any]] = None, created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRang = None, last_updated_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRang = None, data_set_ids: List[Dict[str, Any]] = None, cognite_client=None*)
Bases: *cognite.client.data_classes._base.CogniteFilter*

No description.

Parameters

- **name** (*str*) – Return only sequences with this *exact* name.

- **external_id_prefix** (*str*) – Filter by this (case-sensitive) prefix for the external ID.
- **metadata** (*Dict[str, Any]*) – Filter the sequences by metadata fields and values (case-sensitive). Format is {"key1":"value1","key2":"value2"}.
- **asset_ids** (*List[int]*) – Return only sequences linked to one of the specified assets.
- **root_asset_ids** (*List[int]*) – Only include sequences that have a related asset in a tree rooted at any of these root assetIds.
- **asset_subtree_ids** (*List[Dict[str, Any]]*) – Only include sequences that have a related asset in a subtree rooted at any of these assetIds (including the roots given). If the total size of the given subtrees exceeds 100,000 assets, an error will be returned.
- **created_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **last_updated_time** (*Union[Dict[str, Any], TimestampRange]*) – Range between two timestamps.
- **data_set_ids** (*List[Dict[str, Any]]*) – Only include sequences that belong to these datasets.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.sequences.**SequenceList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.sequences.**SequenceUpdate**(*id: int = None, external_id: str = None*)

Bases: *cognite.client.data_classes._base.CogniteUpdate*

No description.

Parameters

- **id** (*int*) – A server-generated ID for the object.
- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.

2.4.10 Raw

Databases

List databases

RawDatabasesAPI.**list**(*limit: int = 25*) → cognite.client.data_classes.raw.DatabaseList

List databases

Parameters **limit** (*int, optional*) – Maximum number of databases to return. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns List of requested databases.

Return type *DatabaseList*

Examples

List the first 5 databases:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> db_list = c.raw.databases.list(limit=5)
```

Iterate over databases:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for db in c.raw.databases:
...     db # do something with the db
```

Iterate over chunks of databases to reduce memory load:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for db_list in c.raw.databases(chunk_size=2500):
...     db_list # do something with the dbs
```

Create new databases

RawDatabasesAPI.**create**(*name: Union[str, List[str]]*) → Union[cognite.client.data_classes.raw.Database, cognite.client.data_classes.raw.DatabaseList]

Create one or more databases.

Parameters **name** (*Union[str, List[str]]*) – A db name or list of db names to create.

Returns Database or list of databases that has been created.

Return type Union[*Database*, *DatabaseList*]

Examples

Create a new database:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.raw.databases.create("db1")
```

Delete databases

RawDatabasesAPI.**delete**(*name: Union[str, List[str]], recursive: bool = False*) → None

Delete one or more databases.

Parameters

- **name** (*Union[str, List[str]]*) – A db name or list of db names to delete.
- **recursive** (*bool*) – Recursively delete all tables in the database(s).

Returns None

Examples

Delete a list of databases:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.raw.databases.delete(["db1", "db2"])
```

Tables

List tables in a database

RawTablesAPI.**list**(*db_name: str*, *limit: int = 25*) → cognite.client.data_classes.raw.TableList

List tables

Parameters

- **db_name** (*str*) – The database to list tables from.
- **limit** (*int, optional*) – Maximum number of tables to return. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns List of requested tables.

Return type *TableList*

Examples

List the first 5 tables:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> table_list = c.raw.tables.list("db1", limit=5)
```

Iterate over tables:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for table in c.raw.tables(db_name="db1"):
...     table # do something with the table
```

Iterate over chunks of tables to reduce memory load:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for table_list in c.raw.tables(db_name="db1", chunk_size=2500):
...     table_list # do something with the tables
```

Create new tables in a database

RawTablesAPI.**create**(*db_name: str*, *name: Union[str, List[str]]*) → Union[cognite.client.data_classes.raw.Table, cognite.client.data_classes.raw.TableList]

Create one or more tables.

Parameters

- **db_name** (*str*) – Database to create the tables in.
- **name** (*Union[str, List[str]]*) – A table name or list of table names to create.

Returns Table or list of tables that has been created.

Return type Union[*Table*, *TableList*]

Examples

Create a new table in a database:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.raw.tables.create("db1", "table1")
```

Delete tables from a database

RawTablesAPI.**delete**(*db_name: str, name: Union[str, List[str]]*) → None

Delete one or more tables.

Parameters

- **db_name** (*str*) – Database to delete tables from.
- **name** (*Union[str, List[str]]*) – A table name or list of table names to delete.

Returns None

Examples

Delete a list of tables:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.raw.tables.delete("db1", ["table1", "table2"])
```

Rows

Get a row from a table

RawRowsAPI.**retrieve**(*db_name: str, table_name: str, key: str*) → Optional[cognite.client.data_classes.raw.Row]

Retrieve a single row by key.

Parameters

- **db_name** (*str*) – Name of the database.
- **table_name** (*str*) – Name of the table.
- **key** (*str*) – The key of the row to retrieve.

Returns The requested row.

Return type Optional[*Row*]

Examples

Retrieve a row with key 'k1' from tablew 't1' in database 'db1':

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> row = c.raw.rows.retrieve("db1", "t1", "k1")
```

List rows in a table

RawRowsAPI.**list**(*db_name: str, table_name: str, min_last_updated_time: int = None, max_last_updated_time: int = None, columns: List[str] = None, limit: int = 25*) → cognite.client.data_classes.raw.RowList

List rows in a table.

Parameters

- **db_name** (*str*) – Name of the database.
- **table_name** (*str*) – Name of the table.
- **min_last_updated_time** (*int*) – Rows must have been last updated after this time. ms since epoch.
- **max_last_updated_time** (*int*) – Rows must have been last updated before this time. ms since epoch.
- **columns** (*List[str]*) – List of column keys. Set to *None* for retrieving all, use [] to retrieve only row keys.
- **limit** (*int*) – The number of rows to retrieve. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns The requested rows.

Return type *RowList*

Examples

List rows:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> row_list = c.raw.rows.list("db1", "t1", limit=5)
```

Iterate over rows:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for row in c.raw.rows(db_name="db1", table_name="t1", columns=["col1","col2
↪"]):
...     row # do something with the row
```

Iterate over chunks of rows to reduce memory load:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for row_list in c.raw.rows(db_name="db1", table_name="t1", chunk_size=2500):
...     row_list # do something with the rows
```

Insert rows into a table

RawRowsAPI.**insert**(*db_name: str, table_name: str, row: Union[List[cognite.client.data_classes.raw.Row], cognite.client.data_classes.raw.Row, Dict[KT, VT]], ensure_parent: bool = False*) → None

Insert one or more rows into a table.

Parameters

- **db_name** (*str*) – Name of the database.
- **table_name** (*str*) – Name of the table.
- **row** (*Union[List[Row], Row, Dict]*) – The row(s) to insert
- **ensure_parent** (*bool*) – Create database/table if they don't already exist.

Returns None

Examples

Insert new rows into a table:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> rows = {"r1": {"col1": "val1", "col2": "val1"}, "r2": {"col1": "val2", "col2
↪": "val2"}}
>>> res = c.raw.rows.insert("db1", "table1", rows)
```

Delete rows from a table

RawRowsAPI.**delete**(*db_name: str, table_name: str, key: Union[str, List[str]]*) → None

Delete rows from a table.

Parameters

- **db_name** (*str*) – Name of the database.
- **table_name** (*str*) – Name of the table.
- **key** (*Union[str, List[str]]*) – The key(s) of the row(s) to delete.

Returns None

Examples

Delete rows from table:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> keys_to_delete = ["k1", "k2", "k3"]
>>> c.raw.rows.delete("db1", "table1", keys_to_delete)
```

Data classes

class cognite.client.data_classes.raw.**Database**(*name: str = None*, *cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

A NoSQL database to store customer data.

Parameters

- **name** (*str*) – Unique name of a database.
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

tables(*limit: int = None*) → cognite.client.data_classes.raw.TableList

Get the tables in this database.

Parameters **limit** (*int*) – The number of tables to return.

Returns List of tables in this database.

Return type *TableList*

class cognite.client.data_classes.raw.**DatabaseList**(*resources: List[Any]*, *cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.raw.**Row**(*key: str = None*, *columns: Dict[str, Any] = None*, *last_updated_time: int = None*, *cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

No description.

Parameters

- **key** (*str*) – Unique row key
- **columns** (*Dict[str, Any]*) – Row data stored as a JSON object.
- **last_updated_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

to_pandas()

Convert the instance into a pandas DataFrame.

Returns The pandas DataFrame representing this instance.

Return type pandas.DataFrame

class cognite.client.data_classes.raw.**RowList**(*resources: List[Any]*, *cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

to_pandas()

Convert the instance into a pandas DataFrame.

Returns The pandas DataFrame representing this instance.

Return type pandas.DataFrame

class cognite.client.data_classes.raw.**Table**(*name: str = None*, *cognite_client=None*)
Bases: *cognite.client.data_classes._base.CogniteResource*

A NoSQL database table to store customer data

Parameters

- **name** (*str*) – Unique name of the table
- **cognite_client** (CogniteClient) – The client to associate with this object.

rows(*key: str = None*, *limit: int = None*) → Union[cognite.client.data_classes.raw.Row, cognite.client.data_classes.raw.RowList]
Get the rows in this table.

Parameters

- **key** (*str*) – Specify a key to return only that row.
- **limit** (*int*) – The number of rows to return.

Returns List of tables in this database.

Return type Union[*Row*, *RowList*]

class cognite.client.data_classes.raw.**TableList**(*resources: List[Any]*, *cognite_client=None*)
Bases: *cognite.client.data_classes._base.CogniteResourceList*

2.4.11 3D

Models

Retrieve a model by ID

ThreeDModelsAPI.**retrieve**(*id: int*) → cognite.client.data_classes.three_d.ThreeDModel
Retrieve a 3d model by id

Parameters **id** (*int*) – Get the model with this id.

Returns The requested 3d model.

Return type *ThreeDModel*

Example

Get 3d model by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.three_d.models.retrieve(id=1)
```

List models

ThreeDModelsAPI.**list**(*published: bool = None*, *limit: int = 25*) → cognite.client.data_classes.three_d.ThreeDModelList
List 3d models.

Parameters

- **published** (*bool*) – Filter based on whether or not the model has published revisions.
- **limit** (*int*) – Maximum number of models to retrieve. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns The list of 3d models.

Return type *ThreeDModelList*

Examples

List 3d models:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> three_d_model_list = c.three_d.models.list()
```

Iterate over 3d models:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for three_d_model in c.three_d.models:
...     three_d_model # do something with the 3d model
```

Iterate over chunks of 3d models to reduce memory load:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for three_d_model in c.three_d.models(chunk_size=50):
...     three_d_model # do something with the 3d model
```

Create models

ThreeDModelsAPI.**create**(*name: Union[str, List[str]]*) → Union[cognite.client.data_classes.three_d.ThreeDModel, cognite.client.data_classes.three_d.ThreeDModelList]

Create new 3d models.

Parameters **name** (*Union[str, List[str]]*) – The name of the 3d model(s) to create.

Returns The created 3d model(s).

Return type Union[*ThreeDModel*, *ThreeDModelList*]

Example

Create new 3d models:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.three_d.models.create(name="My Model")
```

Update models

ThreeDModelsAPI.**update**(*item: Union[cognite.client.data_classes.three_d.ThreeDModel, cognite.client.data_classes.three_d.ThreeDModelUpdate, List[Union[cognite.client.data_classes.three_d.ThreeDModel, cognite.client.data_classes.three_d.ThreeDModelList]]]*) → Union[cognite.client.data_classes.three_d.ThreeDModel, cognite.client.data_classes.three_d.ThreeDModelList]

Update 3d models.

Parameters **item** (*Union[ThreeDModel, ThreeDModelUpdate, List[Union[ThreeDModel, ThreeDModelUpdate]]]*) – ThreeDModel(s) to update

Returns Updated ThreeDModel(s)

Return type Union[*ThreeDModel*, *ThreeDModelList*]

Examples

Update 3d model that you have fetched. This will perform a full update of the model:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> three_d_model = c.three_d.models.retrieve(id=1)
>>> three_d_model.name = "New Name"
>>> res = c.three_d.models.update(three_d_model)
```

Perform a partial update on a 3d model:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import ThreeDModelUpdate
>>> c = CogniteClient()
>>> my_update = ThreeDModelUpdate(id=1).name.set("New Name")
>>> res = c.three_d.models.update(my_update)
```

Delete models

ThreeDModelsAPI.**delete**(*id: Union[int, List[int]]*) → None

Delete 3d models.

Parameters **id** (*Union[int, List[int]]*) – ID or list of IDs to delete.

Returns None

Example

Delete 3d model by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.three_d.models.delete(id=1)
```

Revisions

Retrieve a revision by ID

ThreeDRevisionsAPI.**retrieve**(*model_id: int, id: int*) → cognite.client.data_classes.three_d.ThreeDModelRevision

Retrieve a 3d model revision by id

Parameters

- **model_id** (*int*) – Get the revision under the model with this id.
- **id** (*int*) – Get the model revision with this id.

Returns The requested 3d model revision.

Return type *ThreeDModelRevision*

Example

Retrieve 3d model revision by model id and revision id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.three_d.revisions.retrieve(model_id=1, id=1)
```

Create a revision

ThreeDRevisionsAPI.**create**(*model_id: int, revision: Union[cognite.client.data_classes.three_d.ThreeDModelRevision, List[cognite.client.data_classes.three_d.ThreeDModelRevision]]*) → Union[cognite.client.data_classes.three_d.ThreeDModelRevision, cognite.client.data_classes.three_d.ThreeDModelRevisionList]

Create a revisions for a specified 3d model.

Parameters

- **model_id** (*int*) – Create revisions for this model.
- **revision** (*Union[ThreeDModelRevision, List[ThreeDModelRevision]]*) – The revision(s) to create.

Returns The created revision(s)

Return type Union[*ThreeDModelRevision*, *ThreeDModelRevisionList*]

Example

Create 3d model revision:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import ThreeDModelRevision
>>> c = CogniteClient()
>>> my_revision = ThreeDModelRevision(file_id=1)
>>> res = c.three_d.revisions.create(model_id=1, revision=my_revision)
```

List revisions

ThreeDRevisionsAPI.**list**(*model_id: int, published: bool = False, limit: int = 25*) → cognite.client.data_classes.three_d.ThreeDModelRevisionList

List 3d model revisions.

Parameters

- **model_id** (*int*) – List revisions under the model with this id.
- **published** (*bool*) – Filter based on whether or not the revision is published.
- **limit** (*int*) – Maximum number of models to retrieve. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns The list of 3d model revisions.

Return type *ThreeDModelRevisionList*

Example

List 3d model revisions:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.three_d.revisions.list(model_id=1, published=True, limit=100)
```

Update revisions

ThreeDRevisionsAPI.**update**(*model_id: int, item: Union[cognite.client.data_classes.three_d.ThreeDModelRevision, cognite.client.data_classes.three_d.ThreeDModelRevisionUpdate, List[Union[cognite.client.data_classes.three_d.ThreeDModelRevision, cognite.client.data_classes.three_d.ThreeDModelRevisionList]]]*) → Union[cognite.client.data_classes.three_d.ThreeDModelRevision, cognite.client.data_classes.three_d.ThreeDModelRevisionList]

Update 3d model revisions.

Parameters

- **model_id** (*int*) – Update the revision under the model with this id.
- **item** (*Union[ThreeDModelRevision, ThreeDModelRevisionUpdate, List[Union[ThreeDModelRevision, ThreeDModelRevisionUpdate]]]*) – ThreeDModelRevision(s) to update

Returns Updated ThreeDModelRevision(s)

Return type Union[*ThreeDModelRevision*, *ThreeDModelRevisionList*]

Examples

Update a revision that you have fetched. This will perform a full update of the revision:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> revision = c.three_d.revisions.retrieve(model_id=1, id=1)
```

(continues on next page)

(continued from previous page)

```
>>> revision.status = "New Status"
>>> res = c.three_d.revisions.update(model_id=1, item=revision)
```

Perform a partial update on a revision, updating the published property and adding a new field to metadata:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import ThreeDModelRevisionUpdate
>>> c = CogniteClient()
>>> my_update = ThreeDModelRevisionUpdate(id=1).published.set(False).metadata.add(
↪{"key": "value"})
>>> res = c.three_d.revisions.update(model_id=1, item=my_update)
```

Delete revisions

ThreeDRevisionsAPI.**delete**(*model_id: int, id: Union[int, List[int]]*) → None

Delete 3d model revisions.

Parameters

- **model_id** (*int*) – Delete the revision under the model with this id.
- **id** (*Union[int, List[int]]*) – ID or list of IDs to delete.

Returns None

Example

Delete 3d model revision by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.three_d.revisions.delete(model_id=1, id=1)
```

Update a revision thumbnail

ThreeDRevisionsAPI.**update_thumbnail**(*model_id: int*, *revision_id: int*, *file_id: int*) → None

Update a revision thumbnail.

Parameters

- **model_id** (*int*) – Id of the model.
- **revision_id** (*int*) – Id of the revision.
- **file_id** (*int*) – Id of the thumbnail file in the Files API.

Returns None

Example

Update revision thumbnail:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.three_d.revisions.update_thumbnail(model_id=1, revision_id=1, file_
↪id=1)
```

List nodes

ThreeDRevisionsAPI.**list_nodes**(*model_id: int*, *revision_id: int*, *node_id: int = None*, *depth: int = None*, *limit: int = 25*) → cognite.client.data_classes.three_d.ThreeDNodeList

Retrieves a list of nodes from the hierarchy in the 3D Model.

You can also request a specific subtree with the 'nodeId' query parameter and limit the depth of the resulting subtree with the 'depth' query parameter.

Parameters

- **model_id** (*int*) – Id of the model.
- **revision_id** (*int*) – Id of the revision.
- **node_id** (*int*) – ID of the root node of the subtree you request (default is the root node).
- **depth** (*int*) – Get sub nodes up to this many levels below the specified node. Depth 0 is the root node.
- **limit** (*int*) – Maximun number of nodes to return. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns The list of 3d nodes.

Return type *ThreeDNodeList*

Example

List nodes from the hierarchy in the 3d model:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.three_d.revisions.list_nodes(model_id=1, revision_id=1, limit=10)
```

List ancestor nodes

ThreeDRevisionsAPI.**list_ancestor_nodes**(*model_id: int*, *revision_id: int*, *node_id: int = None*, *limit: int = 25*) → cognite.client.data_classes.three_d.ThreeDNodeList

Retrieves a list of ancestor nodes of a given node, including itself, in the hierarchy of the 3D model

You can also request a specific subtree with the 'nodeId' query parameter and limit the depth of the resulting subtree with the 'depth' query parameter.

Parameters

- **model_id** (*int*) – Id of the model.
- **revision_id** (*int*) – Id of the revision.
- **node_id** (*int*) – ID of the node to get the ancestors of.

- **limit** (*int*) – Maximun number of nodes to return. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns The list of 3d nodes.

Return type *ThreeDNodeList*

Example

Get a list of ancestor nodes of a given node:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.three_d.revisions.list_ancestor_nodes(model_id=1, revision_id=1, node_
↪id=5, limit=10)
```

Files

Retrieve a 3D file

ThreeDFilesAPI.**retrieve**(*id: int*) → bytes

Retrieve the contents of a 3d file by id.

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

Returns The contents of the file.

Return type bytes

Example

Retrieve the contents of a 3d file by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.three_d.files.retrieve(1)
```

Asset mappings

Create an asset mapping

ThreeDAssetMappingAPI.**create**(*model_id: int, revision_id: int, asset_mapping: Union[cognite.client.data_classes.three_d.ThreeDAssetMapping, List[cognite.client.data_classes.three_d.ThreeDAssetMapping]]*) → Union[cognite.client.data_classes.three_d.ThreeDAssetMapping, cognite.client.data_classes.three_d.ThreeDAssetMappingList]

Create 3d node asset mappings.

Parameters

- **model_id** (*int*) – Id of the model.
- **revision_id** (*int*) – Id of the revision.

- **asset_mapping** (*Union[ThreeDAssetMapping, List[ThreeDAssetMapping]]*) – The asset mapping(s) to create.

Returns The created asset mapping(s).

Return type Union[*ThreeDAssetMapping*, *ThreeDAssetMappingList*]

Example

Create new 3d node asset mapping:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import ThreeDAssetMapping
>>> my_mapping = ThreeDAssetMapping(node_id=1, asset_id=1)
>>> c = CogniteClient()
>>> res = c.three_d.asset_mappings.create(model_id=1, revision_id=1, asset_
↪mapping=my_mapping)
```

List asset mappings

ThreeDAssetMappingAPI.**list**(*model_id: int*, *revision_id: int*, *node_id: int = None*, *asset_id: int = None*, *limit: int = 25*) → cognite.client.data_classes.three_d.ThreeDAssetMappingList

List 3D node asset mappings.

Parameters

- **model_id** (*int*) – Id of the model.
- **revision_id** (*int*) – Id of the revision.
- **node_id** (*int*) – List only asset mappings associated with this node.
- **asset_id** (*int*) – List only asset mappings associated with this asset.
- **limit** (*int*) – Maximum number of asset mappings to return. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns The list of asset mappings.

Return type *ThreeDAssetMappingList*

Example

List 3d node asset mappings:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.three_d.asset_mappings.list(model_id=1, revision_id=1)
```

Delete asset mappings

ThreeDAssetMappingAPI.**delete**(*model_id: int*, *revision_id: int*, *asset_mapping: Union[cognite.client.data_classes.three_d.ThreeDAssetMapping, List[cognite.client.data_classes.three_d.ThreeDAssetMapping]]*) → None

Delete 3d node asset mappings.

Parameters

- **model_id** (*int*) – Id of the model.
- **revision_id** (*int*) – Id of the revision.
- **asset_mapping** (*Union[ThreeDAssetMapping, List[ThreeDAssetMapping]]*) – The asset mapping(s) to delete.

Returns None

Example

Delete 3d node asset mapping:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> mapping_to_delete = c.three_d.asset_mappings.list(model_id=1, revision_
↪id=1)[0]
>>> res = c.three_d.asset_mappings.delete(model_id=1, revision_id=1, asset_
↪mapping=mapping_to_delete)
```

Data classes

class cognite.client.data_classes.three_d.**BoundingBox3D**(*max: List[float] = None, min: List[float] = None, **kwargs*)

Bases: dict

The bounding box of the subtree with this sector as the root sector. Is null if there are no geometries in the subtree.

Parameters

- **max** (*List[float]*) – No description.
- **min** (*List[float]*) – No description.

class cognite.client.data_classes.three_d.**RevisionCameraProperties**(*target: List[float] = None, position: List[float] = None, **kwargs*)

Bases: dict

Initial camera position and target.

Parameters

- **target** (*List[float]*) – Initial camera target.
- **position** (*List[float]*) – Initial camera position.

class cognite.client.data_classes.three_d.**ThreeDAssetMapping**(*node_id: int = None, asset_id: int = None, tree_index: int = None, subtree_size: int = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

No description.

Parameters

- **node_id** (*int*) – The ID of the node.
- **asset_id** (*int*) – The ID of the associated asset (Cognite's Assets API).
- **tree_index** (*int*) – A number describing the position of this node in the 3D hierarchy, starting from 0. The tree is traversed in a depth-first order.
- **subtree_size** (*int*) – The number of nodes in the subtree of this node (this number included the node itself).
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.three_d.**ThreeDAssetMappingList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.three_d.**ThreeDModel**(*name: str = None, id: int = None, created_time: int = None, metadata: Dict[str, str] = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

No description.

Parameters

- **name** (*str*) – The name of the model.
- **id** (*int*) – The ID of the model.
- **created_time** (*int*) – The creation time of the resource, in milliseconds since January 1, 1970 at 00:00 UTC.
- **metadata** (*Dict[str, str]*) – Custom, application specific metadata. String key -> String value. Limits: Maximum length of key is 32 bytes, value 512 bytes, up to 16 key-value pairs.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.three_d.**ThreeDModelList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.three_d.**ThreeDModelRevision**(*id: int = None, file_id: int = None, published: bool = None, rotation: List[float] = None, camera: Union[Dict[str, Any], cognite.client.data_classes.three_d.RevisionC = None, status: str = None, metadata: Dict[str, str] = None, thumbnail_threed_file_id: int = None, thumbnail_url: str = None, asset_mapping_count: int = None, created_time: int = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

No description.

Parameters

- **id** (*int*) – The ID of the revision.
- **file_id** (*int*) – The file id.
- **published** (*bool*) – True if the revision is marked as published.
- **rotation** (*List[float]*) – No description.
- **camera** (*Union[Dict[str, Any], RevisionCameraProperties]*) – Initial camera position and target.
- **status** (*str*) – The status of the revision.
- **metadata** (*Dict[str, str]*) – Custom, application specific metadata. String key -> String value. Limits: Maximum length of key is 32 bytes, value 512 bytes, up to 16 key-value pairs.
- **thumbnail_threed_file_id** (*int*) – The threed file ID of a thumbnail for the revision. Use /3d/files/{id} to retrieve the file.
- **thumbnail_url** (*str*) – The URL of a thumbnail for the revision.
- **asset_mapping_count** (*int*) – The number of asset mappings for this revision.
- **created_time** (*int*) – The creation time of the resource, in milliseconds since January 1, 1970 at 00:00 UTC.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.three_d.**ThreeDModelRevisionList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.three_d.**ThreeDModelRevisionUpdate**(*id: int = None, external_id: str = None*)

Bases: *cognite.client.data_classes._base.CogniteUpdate*

No description.

Parameters **id** (*int*) – A server-generated ID for the object.

class cognite.client.data_classes.three_d.**ThreeDModelUpdate**(*id: int = None, external_id: str = None*)

Bases: *cognite.client.data_classes._base.CogniteUpdate*

No description.

Parameters **id** (*int*) – A server-generated ID for the object.

class cognite.client.data_classes.three_d.**ThreeDNode**(*id: int = None, tree_index: int = None, parent_id: int = None, depth: int = None, name: str = None, subtree_size: int = None, properties: Dict[str, Dict[str, str]] = None, bounding_box: Union[Dict[str, Any], cognite.client.data_classes.three_d.BoundingBox3D] = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

No description.

Parameters

- **id** (*int*) – The ID of the node.
- **tree_index** (*int*) – The index of the node in the 3D model hierarchy, starting from 0. The tree is traversed in a depth-first order.
- **parent_id** (*int*) – The parent of the node, null if it is the root node.
- **depth** (*int*) – The depth of the node in the tree, starting from 0 at the root node.
- **name** (*str*) – The name of the node.
- **subtree_size** (*int*) – The number of descendants of the node, plus one (counting itself).
- **properties** (*Dict[str, Dict[str, str]]*) – Properties extracted from 3D model, with property categories containing key/value string pairs.
- **bounding_box** (*Union[Dict[str, Any], BoundingBox3D]*) – The bounding box of the subtree with this sector as the root sector. Is null if there are no geometries in the subtree.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.three_d.**ThreeDNodeList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

2.4.12 Identity and access management

Service accounts

List service accounts

ServiceAccountsAPI.**list**() → cognite.client.data_classes.iam.ServiceAccountList

List service accounts.

Returns List of service accounts.

Return type *ServiceAccountList*

Example

List service accounts:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.iam.service_accounts.list()
```

Create service accounts

ServiceAccountsAPI.**create**(*service_account: Union[cognite.client.data_classes.iam.ServiceAccount, List[cognite.client.data_classes.iam.ServiceAccount]]*) → Union[cognite.client.data_classes.iam.ServiceAccount, cognite.client.data_classes.iam.ServiceAccountList]

Create one or more new service accounts.

Parameters **service_account** (*Union[ServiceAccount, List[ServiceAccount]]*) – The service account(s) to create.

Returns The created service account(s).

Return type Union[*ServiceAccount*, *ServiceAccountList*]

Example

Create service account:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import ServiceAccount
>>> c = CogniteClient()
>>> my_account = ServiceAccount(name="my@service.com", groups=[1, 2, 3])
>>> res = c.iam.service_accounts.create(my_account)
```

Delete service accounts

ServiceAccountsAPI.**delete**(*id: Union[int, List[int]]*) → None

Delete one or more service accounts.

Parameters **id** (*Union[int, List[int]]*) – ID or list of IDs to delete.

Returns None

Example

Delete service account by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.iam.service_accounts.delete(1)
```

API keys

List API keys

APIKeysAPI.**list**(*include_deleted: bool = False*, *all: bool = False*, *service_account_id: bool = None*) → cognite.client.data_classes.iam.APIKeyList

List api keys.

Parameters

- **include_deleted** (*bool*) – Whether or not to include deleted api keys. Defaults to False.
- **all** (*bool*) – Whether or not to return all api keys for this project. Requires users:list acl. Defaults to False.
- **service_account_id** (*int*) – Get api keys for this service account only. Only available to admin users.

Returns List of api keys.

Return type *APIKeyList*

Example

List api keys:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.iam.api_keys.list()
```

Create API keys

APIKeysAPI.**create**(*service_account_id: Union[int, List[int]]*) → Union[cognite.client.data_classes.iam.APIKey, cognite.client.data_classes.iam.APIKeyList]

Create a new api key for one or more service accounts.

Parameters **service_account_id** (*Union[int, List[int]]*) – ID or list of IDs of service accounts to create an api key for.

Returns API key or list of api keys.

Return type Union[*APIKey*, *APIKeyList*]

Example

Create new api key for a given service account:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.iam.api_keys.create(1)
```

Delete API keys

APIKeysAPI.**delete**(*id: Union[int, List[int]]*) → None
Delete one or more api keys.

Parameters **id** (*Union[int, List[int]]*) – ID or list of IDs of api keys to delete.

Returns None

Example

Delete api key for a given service account:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.iam.api_keys.delete(1)
```

Groups

List groups

GroupsAPI.**list**(*all: bool = False*) → cognite.client.data_classes.iam.GroupList
List groups.

Parameters **all** (*bool*) – Whether to get all groups, only available with the groups:list acl.

Returns List of groups.

Return type *GroupList*

Example

List groups:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.iam.groups.list()
```

Create groups

GroupsAPI.**create**(*group: Union[cognite.client.data_classes.iam.Group, List[cognite.client.data_classes.iam.Group]]*) → Union[cognite.client.data_classes.iam.Group, cognite.client.data_classes.iam.GroupList]
Create one or more groups.

Parameters **group** (*Union[Group, List[Group]]*) – Group or list of groups to create.

Returns The created group(s).

Return type Union[*Group*, *GroupList*]

Example

Create group:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import Group
>>> c = CogniteClient()
>>> my_capabilities = [{"groupsAcl": {"actions": ["LIST"],"scope": {"all": { }}}}]
>>> my_group = Group(name="My Group", capabilities=my_capabilities)
>>> res = c.iam.groups.create(my_group)
```

Delete groups

GroupsAPI.**delete**(*id: Union[int, List[int]]*) → None

Delete one or more groups.

Parameters **id** (*Union[int, List[int]]*) – ID or list of IDs of groups to delete.

Returns None

Example

Delete group:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.iam.groups.delete(1)
```

List service accounts in a group

GroupsAPI.**list_service_accounts**(*id: int*) → cognite.client.data_classes.iam.ServiceAccountList

List service accounts in a group.

Parameters **id** (*int*) – List service accounts which are a member of this group.

Returns List of service accounts.

Return type *ServiceAccountList*

Example

List service accounts in a group:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.iam.groups.list_service_accounts(1)
```

Add service accounts to a group

GroupsAPI.**add_service_account**(*id: int, service_account_id: Union[int, List[int]]*) → None
Add one or more service accounts to a group.

Parameters

- **id** (*int*) – Add service accounts to the group with this id.
- **service_account_id** (*Union[int, List[int]]*) – Add these service accounts to the specified group.

Returns None

Example

Add service account to group:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.iam.groups.add_service_account(id=1, service_account_id=1)
```

Remove service accounts from a group

GroupsAPI.**remove_service_account**(*id: int, service_account_id: Union[int, List[int]]*) → None
Remove one or more service accounts from a group.

Parameters

- **id** (*int*) – Remove service accounts from the group with this id.
- **service_account_id** – Remove these service accounts from the specified group.

Returns None

Example

Remove service account from group:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.iam.groups.remove_service_account(id=1, service_account_id=1)
```

Security categories

List security categories

SecurityCategoriesAPI.**list**(*limit: int = 25*) → cognite.client.data_classes.iam.SecurityCategoryList
List security categories.

Parameters **limit** (*int*) – Max number of security categories to return. Defaults to 25.

Returns List of security categories

Return type *SecurityCategoryList*

Example

List security categories:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.iam.security_categories.list()
```

Create security categories

SecurityCategoriesAPI.**create**(*security_category: Union[cognite.client.data_classes.iam.SecurityCategory, List[cognite.client.data_classes.iam.SecurityCategory]]*) → Union[cognite.client.data_classes.iam.SecurityCategory, cognite.client.data_classes.iam.SecurityCategoryList]

Create one or more security categories.

Parameters **security_category** (*Union[SecurityCategory, List[SecurityCategory]]*) – Security category or list of categories to create.

Returns The created security category or categories.

Return type Union[*SecurityCategory*, *SecurityCategoryList*]

Example

Create security category:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import SecurityCategory
>>> c = CogniteClient()
>>> my_category = SecurityCategory(name="My Category")
>>> res = c.iam.security_categories.create(my_category)
```

Delete security categories

SecurityCategoriesAPI.**delete**(*id: Union[int, List[int]]*) → None

Delete one or more security categories.

Parameters **id** (*Union[int, List[int]]*) – ID or list of IDs of security categories to delete.

Returns None

Example

Delete security category:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.iam.security_categories.delete(1)
```

Data classes

class cognite.client.data_classes.iam.**APIKey**(*id: int = None, service_account_id: int = None, created_time: int = None, status: str = None, value: str = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

No description.

Parameters

- **id** (*int*) – The internal ID for the API key.
- **service_account_id** (*int*) – The ID of the service account.
- **created_time** (*int*) – The time of creation in Unix milliseconds.
- **status** (*str*) – The status of the API key.
- **value** (*str*) – The API key to be used against the API.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.iam.**APIKeyList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.iam.**Group**(*name: str = None, source_id: str = None, capabilities: List[Dict[str, Any]] = None, id: int = None, is_deleted: bool = None, deleted_time: int = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

No description.

Parameters

- **name** (*str*) – Name of the group
- **source_id** (*str*) – ID of the group in the source. If this is the same ID as a group in the IDP, a service account in that group will implicitly be a part of this group as well.
- **capabilities** (*List[Dict[str, Any]]*) – No description.
- **id** (*int*) – No description.
- **is_deleted** (*bool*) – No description.
- **deleted_time** (*int*) – No description.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.iam.**GroupList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.iam.**SecurityCategory**(*name: str = None, id: int = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

No description.

Parameters

- **name** (*str*) – Name of the security category
- **id** (*int*) – Id of the security category
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.iam.**SecurityCategoryList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.iam.**ServiceAccount**(*name: str = None, groups: List[int] = None, id: int = None, is_deleted: bool = None, deleted_time: int = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

No description.

Parameters

- **name** (*str*) – Unique name of the service account
- **groups** (*List[int]*) – List of group ids
- **id** (*int*) – No description.
- **is_deleted** (*bool*) – If this service account has been logically deleted
- **deleted_time** (*int*) – Time of deletion
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.iam.**ServiceAccountList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

2.4.13 Base data classes

CogniteResource

class cognite.client.data_classes._base.**CogniteResource**

dump(*camel_case: bool = False*) → Dict[str, Any]

Dump the instance into a json serializable Python data type.

Parameters **camel_case** (*bool*) – Use camelCase for attribute names. Defaults to False.

Returns A dictionary representation of the instance.

Return type Dict[str, Any]

to_pandas(*expand: List[str] = ('metadata',), ignore: List[str] = None, camel_case: bool = True*)

Convert the instance into a pandas DataFrame.

Parameters

- **expand** (*List[str]*) – List of row keys to expand, only works if the value is a Dict. Will expand metadata by default.
- **ignore** (*List[str]*) – List of row keys to not include when converting to a data frame.
- **camel_case** (*bool*) – Convert column names to camel case (e.g. *externalId* instead of *external_id*)

Returns The dataframe.

Return type pandas.DataFrame

CogniteResourceList

class cognite.client.data_classes._base.**CogniteResourceList**(*resources: List[Any], cognite_client=None*)

dump(*camel_case: bool = False*) → List[Dict[str, Any]]
Dump the instance into a json serializable Python data type.

Parameters **camel_case** (*bool*) – Use camelCase for attribute names. Defaults to False.

Returns A list of dicts representing the instance.

Return type List[Dict[str, Any]]

get(*id: int = None, external_id: str = None*) → Optional[cognite.client.data_classes._base.CogniteResource]
Get an item from this list by id or exernal_id.

Parameters

- **id** (*int*) – The id of the item to get.
- **external_id** (*str*) – The external_id of the item to get.

Returns The requested item

Return type Optional[*CogniteResource*]

to_pandas(*camel_case=True*) → pandas.DataFrame
Convert the instance into a pandas DataFrame.

Returns The dataframe.

Return type pandas.DataFrame

CogniteResponse

class cognite.client.data_classes._base.**CogniteResponse**

dump(*camel_case: bool = False*) → Dict[str, Any]
Dump the instance into a json serializable python data type.

Parameters **camel_case** (*bool*) – Use camelCase for attribute names. Defaults to False.

Returns A dictionary representation of the instance.

Return type Dict[str, Any]

CogniteFilter

class cognite.client.data_classes._base.**CogniteFilter**

dump(*camel_case: bool = False*)
Dump the instance into a json serializable Python data type.

Returns A dictionary representation of the instance.

Return type Dict[str, Any]

CogniteUpdate

class cognite.client.data_classes._base.**CogniteUpdate**(*id: int = None*, *external_id: str = None*)

dump()
Dump the instance into a json serializable Python data type.

Returns A dictionary representation of the instance.

Return type Dict[str, Any]

2.4.14 Exceptions

CogniteAPIError

exception cognite.client.exceptions.**CogniteAPIError**(*message: str*, *code: int = None*, *x_request_id: str = None*, *missing: List[T] = None*, *duplicated: List[T] = None*, *successful: List[T] = None*, *failed: List[T] = None*, *unknown: List[T] = None*, *unwrap_fn: Callable = None*, *extra: Dict[KT, VT] = None*)

Cognite API Error

Raised if a given request fails. If one or more of concurrent requests fails, this exception will also contain information about which items were successfully processed (2xx), which may have been processed (5xx), and which have failed to be processed (4xx).

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.
- **successful** (*List*) – List of items which were successfully proccessed.
- **failed** (*List*) – List of items which failed.
- **unknown** (*List*) – List of items which may or may not have been successfully processed.

Examples

Catching an API-error and handling it based on the error code:

```
from cognite.client import CogniteClient
from cognite.client.exceptions import CogniteAPIError

c = CogniteClient()
```

(continues on next page)

(continued from previous page)

```
try:
    c.login.status()
except CogniteAPIError as e:
    if e.code == 401:
        print("You are not authorized")
    elif e.code == 400:
        print("Something is wrong with your request")
    elif e.code == 500:
        print("Something went terribly wrong. Here is the request-id: {}".
↪format(e.x_request_id)
    print("The message returned from the API: {}".format(e.message))
```

CogniteNotFoundError

exception cognite.client.exceptions.**CogniteNotFoundError**(*not_found: List[T], successful: List[T] = None, failed: List[T] = None, unknown: List[T] = None, unwrap_fn: Callable = None*)

Cognite Not Found Error

Raised if one or more of the referenced ids/external ids are not found.

Parameters

- **not_found** (*List*) – The ids not found.
- **successful** (*List*) – List of items which were successfully proccessed.
- **failed** (*List*) – List of items which failed.
- **unknown** (*List*) – List of items which may or may not have been successfully processed.

CogniteDuplicatedError

exception cognite.client.exceptions.**CogniteDuplicatedError**(*duplicated: List[T], successful: List[T] = None, failed: List[T] = None, unknown: List[T] = None, unwrap_fn: Callable = None*)

Cognite Duplicated Error

Raised if one or more of the referenced ids/external ids have been duplicated in the request.

Parameters

- **duplicated** (*list*) – The duplicated ids.
- **successful** (*List*) – List of items which were successfully proccessed.
- **failed** (*List*) – List of items which failed.
- **unknown** (*List*) – List of items which may or may not have been successfully processed.

CogniteAPIKeyError

exception cognite.client.exceptions.**CogniteAPIKeyError**
Cognite API Key Error.

Raised if the API key is missing or invalid.

CogniteImportError

exception cognite.client.exceptions.**CogniteImportError**(*module: str*, *message: str = None*)
Cognite Import Error

Raised if the user attempts to use functionality which requires an uninstalled package.

Parameters

- **module** (*str*) – Name of the module which could not be imported
- **message** (*str*) – The error message to output.

CogniteMissingClientError

exception cognite.client.exceptions.**CogniteMissingClientError**
Cognite Missing Client Error

Raised if the user attempts to make use of a method which requires the cognite_client being set, but it is not.

CogniteDuplicateColumnsError

exception cognite.client.exceptions.**CogniteDuplicateColumnsError**(*dups*)
Cognite Duplicate Columns Error

Raised if the user attempts to create a dataframe through include_aggregate_names=False which results in duplicate column names.

2.4.15 Utils

Convert timestamp to milliseconds since epoch

cognite.client.utils.**timestamp_to_ms**(*timestamp: Union[int, float, str, datetime.datetime]*) → int
Returns the ms representation of some timestamp given by milliseconds, time-ago format or datetime object

Parameters **timestamp** (*Union[int, float, str, datetime]*) – Convert this timestamp to ms.

Returns Milliseconds since epoch representation of timestamp

Return type int

Convert milliseconds since epoch to datetime

cognite.client.utils.**ms_to_datetime**(*ms: Union[int, float]*) → datetime.datetime
Converts milliseconds since epoch to datetime object.

Parameters **ms** (*Union[int, float]*) – Milliseconds since epoch

Returns Datetime object.

Return type datetime

2.4.16 Testing

Object to use as a mock for CogniteClient

class cognite.client.testing.**CogniteClientMock**(**args*, ***kwargs*)
Mock for CogniteClient object

All APIs are replaced with specced MagicMock objects.

Use a context manager to monkeypatch CogniteClient

cognite.client.testing.**monkeypatch_cognite_client**()
Context manager for monkeypatching the CogniteClient.

Will patch all clients and replace them with specced MagicMock objects.

Yields *CogniteClientMock* – The mock with which the CogniteClient has been replaced

Examples

In this example we can run the following code without actually executing the underlying API calls:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TimeSeries
>>> from cognite.client.testing import monkeypatch_cognite_client
>>>
>>> with monkeypatch_cognite_client():
>>>     c = CogniteClient()
>>>     c.time_series.create(TimeSeries(external_id="blabla"))
```

This example shows how to set the return value of a given method:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TimeSeries
>>> from cognite.client.data_classes import LoginStatus
>>> from cognite.client.testing import monkeypatch_cognite_client
>>>
>>> with monkeypatch_cognite_client() as c_mock:
>>>     c_mock.login.status.return_value = LoginStatus(
>>>         user="user", project="dummy", project_id=1, logged_in=True, api_key_
↪id=1
>>>     )
>>>     c = CogniteClient()
>>>     res = c.login.status()
>>>     assert "user" == res.user
```

Here you can see how to have a given method raise an exception:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.exceptions import CogniteAPIError
>>> from cognite.client.testing import monkeypatch_cognite_client
>>>
>>> with monkeypatch_cognite_client() as c_mock:
>>>     c_mock.login.status.side_effect = CogniteAPIError(message="Something went␣
↪wrong", code=400)
>>>     c = CogniteClient()
>>>     try:
>>>         res = c.login.status()
>>>     except CogniteAPIError as e:
>>>         assert 400 == e.code
>>>         assert "Something went wrong" == e.message
```

2.5 Experimental features

Warning: These features are subject to breaking changes and should not be used in production code.

2.5.1 Model Hosting

Warning: The model hosting API is experimental and subject to breaking changes. It should not be used in production code.

Models

Retrieve model by name

ModelsAPI.**get_model**(*name: str*) → cognite.client.data_classes.model_hosting.models.Model

Get a model by name.

Parameters **name** (*str*) – Name of model to get.

Returns The requested model

Return type *Model*

List models

ModelsAPI.**list_models**(*limit: int = None*, *cursor: int = None*, *autopaging: bool = False*) → cognite.client.data_classes.model_hosting.models.ModelList

List all models.

Parameters

- **limit** (*int*) – Maximum number of models to return. Defaults to 250.
- **cursor** (*str*) – Cursor to use to fetch next set of results.
- **autopaging** (*bool*) – Whether or not to automatically page through all results. Will disregard limit.

Returns List of models

Return type *ModelList*

Create model

ModelsAPI.**create_model**(*name: str*, *description: str = ''*, *metadata: Dict[str, Any] = None*, *input_fields: List[Dict[str, str]] = None*, *output_fields: List[Dict[str, str]] = None*, *webhook_url: str = None*) → cognite.client.data_classes.model_hosting.models.Model

Creates a new model

Parameters

- **name** (*str*) – Name of model
- **description** (*str*) – Description
- **metadata** (*Dict[str, Any]*) – Metadata about model
- **input_fields** (*List[str]*) – List of input fields the model accepts
- **output_fields** (*List[str]*) – List of output fields the model produces
- **webhook_url** (*str*) – Webhook url to send notifications to upon failing scheduled predictions

Returns The created model.

Return type *Model*

Update model

ModelsAPI.**update_model**(*name: str*, *description: str = None*, *metadata: Dict[str, str] = None*, *active_version_name: int = None*, *webhook_url: str = None*) → cognite.client.data_classes.model_hosting.models.Model

Update a model.

Parameters

- **name** (*str*) – Name of model to update.
- **description** (*str*) – Description of model.
- **metadata** (*Dict[str, str]*) – metadata about model.
- **active_version_name** (*str*) – Active version of model.
- **webhook_url** (*str*) – Webhook url to send notifications to upon failing scheduled predictions.

Returns Updated model

Return type *Model*

Deprecate model

ModelsAPI.**deprecate_model**(*name: str*) → cognite.client.data_classes.model_hosting.models.Model

Deprecate a model.

Parameters **name** (*str*) – Name of model to deprecate.

Returns Deprecated model

Return type *Model*

Delete model

ModelsAPI.**delete_model**(*name: str*) → None

Delete a model.

Will also delete all versions and schedules for this model.

Parameters **name** (*str*) – Delete model with this name.

Returns None

Perform online prediction

ModelsAPI.**online_predict**(*model_name: str*, *version_name: str = None*, *instances: List[T] = None*, *args: Dict[str, Any] = None*) → List[T]

Perform online prediction on a models active version or a specified version.

Parameters

- **model_name** (*str*) – Perform a prediction on the model with this name. Will use active version.
- **version_name** (*str*) – Use this version instead of the active version. (optional)
- **instances** (*List*) – List of JSON serializable instances to pass to your model one-by-one.
- **args** (*Dict[str, Any]*) –

Returns List of predictions for each instance.

Return type List

Model Versions

Retrieve model version by name

ModelVersionsAPI.**get_model_version**(*model_name: str*, *version_name: str*) → cognite.client.data_classes.model_hosting.versions.ModelVersion

Get a specific model version by name.

Parameters

- **model_name** (*str*) – Name of model which has the model version.
- **version_name** (*str*) – Name of model version.

Returns The requested model version

Return type *ModelVersion*

List model versions

ModelVersionsAPI.**list_model_versions**(*model_name: str*, *limit: int = None*, *cursor: str = None*, *autopaging: bool = False*) → cognite.client.data_classes.model_hosting.versions.ModelVersionList

Get all versions of a specific model.

Parameters

- **model_name** (*str*) – Get versions for the model with this name.
- **limit** (*int*) – Maximum number of model versions to return. Defaults to 250.
- **cursor** (*str*) – Cursor to use to fetch next set of results.
- **autopaging** (*bool*) – Whether or not to automatically page through all results. Will disregard limit.

Returns List of model versions

Return type *ModelVersionList*

Create and deploy model version

ModelVersionsAPI.**deploy_model_version**(*model_name: str*, *version_name: str*, *source_package_id: int*, *artifacts_directory: str = None*, *description: str = None*, *metadata: Dict[KT, VT] = None*) → cognite.client.data_classes.model_hosting.versions.ModelVersion

This will create and deploy a model version.

If artifacts_directory is specified, it will traverse that directory recursively and upload all artifacts in that directory before deploying.

Parameters

- **model_name** (*str*) – Create the version on the model with this name.
- **version_name** (*str*) – Name of the the model version.
- **source_package_id** (*int*) – Use the source package with this id. The source package must have an available predict operation.
- **artifacts_directory** (*str, optional*) – Absolute path of directory containing artifacts.
- **description** (*str, optional*) – Description of model version
- **metadata** (*Dict[str, Any], optional*) – Metadata about model version

Returns The created model version.

Return type *ModelVersion*

Create model version without deploying

ModelVersionsAPI.**create_model_version**(*model_name: str*, *version_name: str*, *source_package_id: int*, *description: str = None*, *metadata: Dict[KT, VT] = None*) → cognite.client.data_classes.model_hosting.versions.ModelVersion

Create a model version without deploying it.

Then you can optionally upload artifacts to the model version and later deploy it.

Parameters

- **model_name** (*str*) – Create the version on the model with this name.
- **version_name** (*str*) – Name of the the model version.
- **source_package_id** (*int*) – Use the source package with this id. The source package must have an available predict operation.
- **description** (*str*) – Description of model version
- **metadata** (*Dict[str, Any]*) – Metadata about model version

Returns The created model version.

Return type *ModelVersion*

Deploy awaiting model version

ModelVersionsAPI.**deploy_awaiting_model_version**(*model_name: str*, *version_name: str*) → cognite.client.data_classes.model_hosting.versions.ModelVersion

Deploy an already created model version awaiting manual deployment.

The model version must have status AWAITING_MANUAL_DEPLOYMENT in order for this to work.

Parameters

- **model_name** (*str*) – The name of the model containing the version to deploy.
- **version_name** (*str*) – The name of the model version to deploy.

Returns The deployed model version.

Return type *ModelVersion*

Update model version

ModelVersionsAPI.**update_model_version**(*model_name: str*, *version_name: str*, *description: str = None*, *metadata: Dict[str, str] = None*) → cognite.client.data_classes.model_hosting.versions.ModelVersion

Update description or metadata on a model version.

Parameters

- **model_name** (*str*) – Name of model containing the model version.
- **version_name** (*str*) – Name of model version to update.
- **description** (*str*) – New description.
- **metadata** (*Dict[str, str]*) – New metadata

Returns The updated model version.

Return type *ModelVersion*

Deprecate model version

ModelVersionsAPI.**deprecate_model_version**(*model_name: str*, *version_name: str*) → cognite.client.data_classes.model_hosting.versions.ModelVersion

Deprecate a model version

Parameters

- **model_name** (*str*) – Name of model
- **version_name** (*str*) – name of model version to deprecate

Returns The deprecated model version

Return type *ModelVersion*

Delete model version

ModelVersionsAPI.**delete_model_version**(*model_name: str*, *version_name: str*) → None

Delete a model version by id.

Parameters

- **model_name** (*str*) – Name of model which has the model version.
- **version_name** (*str*) – Name of model version.

Returns None

Model Version Artifacts

List artifacts for a model version

ModelVersionsAPI.**list_artifacts**(*model_name: str*, *version_name: str*) → cognite.client.data_classes.model_hosting.versions.ModelArtifactList

List the artifacts associated with the specified model version.

Parameters

- **model_name** (*str*) – Name of model
- **version_name** (*str*) – Name of model version to get artifacts for

Returns List of artifacts

Return type *ModelArtifactList*

Upload an artifact from a file to a model version awating deployment

ModelVersionsAPI.**upload_artifact_from_file**(*model_name: str*, *version_name: str*, *artifact_name: str*, *file_path: str*) → None

Upload an artifact to a model version.

The model version must have status AWAITING_MANUAL_DEPLOYMENT in order for this to work.

Parameters

- **model_name** (*str*) – The name of the model.
- **version_name** (*str*) – The name of the model version to upload the artifacts to.

- **artifact_name** (*str*) – The name of the artifact.
- **file_path** (*str*) – The local path of the artifact.

Returns None

Upload artifacts from a directory to a model version awating deployment

ModelVersionsAPI.**upload_artifacts_from_directory**(*model_name: str*, *version_name: str*, *directory: str*) → None

Upload all files in directory recursively.

Parameters

- **model_name** (*str*) – The name of the model.
- **version_name** (*str*) – The name of the model version to upload the artifacts to.
- **directory** (*str*) – Absolute path of directory to upload artifacts from.

Returns None

Download an artifact for a model version

ModelVersionsAPI.**download_artifact**(*model_name: str*, *version_name: str*, *artifact_name: str*, *directory: str = None*) → None

Download an artifact to a directory. Defaults to current working directory.

Parameters

- **model_name** (*str*) – Name of model
- **version_name** (*str*) – Name of model version.
- **artifact_name** (*str*) – Name of artifact.
- **directory** (*str*) – Directory to place artifact in. Defaults to current working directory.

Returns None

Schedules

Retrieve schedule by name

SchedulesAPI.**get_schedule**(*model_name: str*, *schedule_name: str*) → cognite.client.data_classes.model_hosting.schedules.Schedule

Get a schedule by name.

Parameters

- **model_name** (*str*) – Name of model associated with this schedule.
- **schedule_name** (*str*) – Name of schedule to get.

Returns The requested schedule.

Return type *Schedule*

List schedules

SchedulesAPI.**list_schedules**(*model_name: str*, *limit: int = None*, *cursor: int = None*, *autopaging: bool = False*) → cognite.client.data_classes.model_hosting.schedules.ScheduleList

Get all schedules.

Parameters

- **model_name** (*str*) – Model for which to list the schedules.
- **limit** (*int*) – Maximum number of schedules to return. Defaults to 250.
- **cursor** (*str*) – Cursor to use to fetch next set of results.
- **autopaging** (*bool*) – Whether or not to automatically page through all results. Will disregard limit.

Returns The requested schedules.

Return type *ScheduleList*

Create Schedule

SchedulesAPI.**create_schedule**(*model_name: str*, *schedule_name: str*, *schedule_data_spec: Any*, *description: str = None*, *args: Dict[KT, VT] = None*, *metadata: Dict[KT, VT] = None*) → cognite.client.data_classes.model_hosting.schedules.Schedule

Create a new schedule on a given model.

Parameters

- **model_name** (*str*) – Name of the model to create schedule on
- **schedule_name** (*str*) – Name of the schedule
- **schedule_data_spec** (*Any*) – Specification of schedule input/output. Can be either a dictionary or a ScheduleDataSpec object from the cognite-model-hosting library.
- **description** (*str*) – Description for schedule
- **args** (*Dict*) – Dictionary of keyword arguments to pass to predict method.
- **metadata** (*Dict*) – Dictionary of metadata about schedule

Returns The created schedule.

Return type *Schedule*

Deprecate Schedule

SchedulesAPI.**deprecate_schedule**(*model_name: str*, *schedule_name: str*) → cognite.client.data_classes.model_hosting.schedules.Schedule

Deprecate a schedule.

Parameters

- **model_name** (*str*) – Name of model associated with this schedule.
- **schedule_name** (*str*) – Name of schedule to deprecate.

Returns The deprecated schedule.

Return type *Schedule*

Delete Schedule

SchedulesAPI.**delete_schedule**(*model_name: str*, *schedule_name: str*) → None

Delete a schedule by id.

Parameters

- **model_name** (*str*) – Name of model associated with this schedule.
- **schedule_name** (*str*) – The name of the schedule to delete.

Returns None

Retrieve schedule logs

SchedulesAPI.**get_log**(*model_name: str*, *schedule_name: str*) → cognite.client.data_classes.model_hosting.schedules.ScheduleLog

Return schedule log by id. The ScheduleLog object contains two logs, one for failed scheduled predictions and one for successful.

Parameters

- **model_name** (*str*) – Name of model associated with this schedule.
- **schedule_name** (*str*) – The name of the schedule to get logs from.

Returns An object containing the schedule logs.

Return type *ScheduleLog*

Source Packages

Retrieve source package by id

SourcePackagesAPI.**get_source_package**(*id: int*) → cognite.client.data_classes.model_hosting.source_packages.SourcePa

Get source package by id.

Parameters **id** (*int*) – Id of soure package to get.

Returns The requested source package.

Return type *SourcePackage*

List source packages

SourcePackagesAPI.**list_source_packages**(*limit: int = None*, *cursor: str = None*, *autopaging: bool = False*) → cognite.client.data_classes.model_hosting.source_packages.SourcePackageList

List all model source packages.

Parameters

- **limit** (*int*) – Maximum number of source_packages to return. Defaults to 250.
- **cursor** (*str*) – Cursor to use to fetch next set of results.

- **autopaging** (*bool*) – Whether or not to automatically page through all results. Will disregard limit.

Returns List of source packages.

Return type *SourcePackageList*

Upload a source package

SourcePackagesAPI.**upload_source_package**(*name: str, package_name: str, available_operations: List[str], runtime_version: str, description: str = None, metadata: Dict[KT, VT] = None, file_path: str = None*) → cognite.client.data_classes.model_hosting.source_packages.CreateSourcePacka

Upload a source package to the model hosting environment.

Parameters

- **name** (*str*) – Name of source package
- **package_name** (*str*) – name of root package for model
- **available_operations** (*List[str]*) – List of routines which this source package supports ["predict", "train"]
- **runtime_version** (*str*) – Version of environment in which the source-package should run. Currently only 0.1.
- **description** (*str*) – Description for source package
- **metadata** (*Dict*) – User defined key value pair of additional information.
- **file_path** (*str*) – File path of source package distribution. If not specified, a download url will be returned.

Returns An response object containing Source package ID if file_path was specified. Else, both source package id and upload url.

Return type *CreateSourcePackageResponse*

Build and upload a source package

SourcePackagesAPI.**build_and_upload_source_package**(*name: str, runtime_version: str, package_directory: str, description: str = None, metadata: Dict[KT, VT] = None*) → cognite.client.data_classes.model_hosting.source_packages.Crea

Build a distribution for a source package and upload it to the model hosting environment.

This method will recursively search through your package and infer available_operations as well as the package name.

Parameters

- **name** (*str*) – Name of source package
- **runtime_version** (*str*) – Version of environment in which the source-package should run. Currently only 0.1.
- **description** (*str*) – Description for source package

- **metadata** (*Dict*) – User defined key value pair of additional information.
- **package_directory** (*str*) – Absolute path of directory containing your setup.py file.

Returns An response object containing Source package ID if file_path was specified. Else, both source package id and upload url.

Return type *CreateSourcePackageResponse*

Deprecate source package

SourcePackagesAPI.**deprecate_source_package**(*id: int*) → cognite.client.data_classes.model_hosting.source_packages.SourcePackage

Deprecate a source package by id.

Parameters **id** (*int*) – Id of soure package to get.

Returns The requested source package.

Return type *SourcePackage*

Delete source package

SourcePackagesAPI.**delete_source_package**(*id: int*) → None

Delete source package by id.

Parameters **id** (*int*) – Id of soure package to delete.

Returns None

Download source package code

SourcePackagesAPI.**download_source_package_code**(*id: int*, *directory: str = None*) → None

Download the tarball for a source package to a specified directory.

Parameters

- **id** (*int*) – Id of source package.
- **directory** (*str*) – Directory to put source package in. Defaults to current working directory.

Returns None

Delete source package code

SourcePackagesAPI.**delete_source_package_code**(*id: int*) → None

Delete the code/tarball for the source package from the cloud storage location. This will only work if the source package has been deprecated.

Warning: This cannot be undone.

Parameters **id** (*int*) – Id of the source package.

Returns None

Data classes

class cognite.client.data_classes.model_hosting.models.**Model**(*name: str = None, description: str = None, created_time: int = None, metadata: Dict[KT, VT] = None, is_deprecated: bool = None, active_version_name: str = None, input_fields: List[T] = None, output_fields: List[T] = None, webhook_url: str = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

A representation of a Model in the model hosting environment.

Parameters

- **name** (*str*) – Name of the model.
- **description** (*str*) – Description of the model.
- **created_time** (*int*) – Created time in UNIX.
- **metadata** (*Dict*) – User-defined metadata about the model.
- **is_deprecated** (*bool*) – Whether or not the model is deprecated.
- **active_version_name** (*str*) – The name of the active version on this model.
- **input_fields** (*List*) – A list of input fields this model takes.
- **output_fields** (*List*) – A list of output fields this model defines.
- **webhook_url** (*str*) – A url used to catch webhooks which are reported upon failing scheduled predictions.
- **cognite_client** (CogniteClient) – An optional CogniteClient to associate with this data class.

class cognite.client.data_classes.model_hosting.models.**ModelList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.model_hosting.versions.**ModelArtifactList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.model_hosting.versions.**ModelVersion**(*name: str = None, is_deprecated: bool = None, training_details: Dict[KT, VT] = None, error_msg: str = None, model_name: str = None, created_time: int = None, metadata: Dict[KT, VT] = None, source_package_id: int = None, status: str = None, description: str = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

A representation of a Model version in the model hosting environment.

Parameters

- **name** (*str*) – Name of the model version.
- **is_deprecated** (*bool*) – Whether or not the model version is deprecated.
- **training_details** (*Dict*) – The training details for this model version. None if the associated source package does not define a .train() method.
- **error_msg** (*str*) – The error message produced when trying to deploy the model version.
- **model_name** (*str*) – The name of the model associated with this version.

- **created_time** (*int*) – Created time in UNIX.
- **metadata** (*Dict*) – User-defined metadata about the model.
- **source_package_id** (*int*) – The id of the source package associated with this version.
- **status** (*str*) – The current status of the model version deployment.
- **description** (*str*) – Description of the model.
- **cognite_client** (CogniteClient) – An optional CogniteClient to associate with this data class.

class cognite.client.data_classes.model_hosting.versions.**ModelVersionList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.model_hosting.versions.**ModelVersionLog**(*prediction_logs: List[T] = None, training_logs: List[T] = None*)

Bases: *cognite.client.data_classes._base.CogniteResponse*

An object containing the logs for a model version.

Parameters

- **prediction_logs** (*List*) – A list of log entries for the prediction routine
- **training_logs** (*List*) – A list of log entries for the training routine

class cognite.client.data_classes.model_hosting.schedules.**LogEntry**(*timestamp: int = None, scheduled_execution_time: int = None, message: str = None*)

Bases: *cognite.client.data_classes._base.CogniteResponse*

An object containing a log entry for a schedule.

Parameters

- **timestamp** (*int*) – The time the log entry was recorded.
- **scheduled_execution_time** (*int*) – The time the prediction was scheduled to run.
- **message** (*str*) – The log message.

class cognite.client.data_classes.model_hosting.schedules.**Schedule**(*name: str = None, model_name: str = None, description: str = None, data_spec: Union[ScheduleDataSpec, Dict[KT, VT]] = None, is_deprecated: bool = None, created_time: int = None, metadata: Dict[KT, VT] = None, args: Dict[KT, VT] = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

A representation of a Schedule in the model hosting environment.

Parameters

- **name** (*str*) – Name of the schedule.
- **model_name** (*str*) – The name of the model associated with this schedule.
- **description** (*str*) – Description of the schedule.
- **data_spec** (*Union[Dict, ScheduleDataSpec]*) – The data spec for the schedule.
- **is_deprecated** (*bool*) – Whether or not the model version is deprecated.
- **created_time** (*int*) – Created time in UNIX.
- **metadata** (*Dict*) – User-defined metadata about the model.
- **args** (*Dict*) – Additional arguments passed to the predict routine.
- **cognite_client** (CogniteClient) – An optional CogniteClient to associate with this data class.

class cognite.client.data_classes.model_hosting.schedules.**ScheduleList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

class cognite.client.data_classes.model_hosting.schedules.**ScheduleLog**(*failed: List[T] = None*, *completed: List[T] = None*)

Bases: *cognite.client.data_classes._base.CogniteResponse*

An object containing the logs for a schedule.

Parameters

- **failed** (*List[LogEntry]*) – A list of log entries for failed executions.
- **completed** (*List[LogEntry]*) – A list of log entries for succesful executions.

class cognite.client.data_classes.model_hosting.source_packages.**CreateSourcePackageResponse**

Bases: *cognite.client.data_classes._base.CogniteResponse*

The response returned from the API when creating a new source package.

Parameters

- **id** (*int*) – The id of the source package
- **upload_url** (*str*) – The url to upload the source package distribution to.

class cognite.client.data_classes.model_hosting.source_packages.**SourcePackage**(*id: int = None, name: str = None, description: str = None, is_deprecated: bool = None, package_name: str = None, is_uploaded: bool = None, available_operations: List[T] = None, created_time: int = None, runtime_version: str = None, metadata: Dict[KT, VT] = None, cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

A representation of a source package in the model hosting environment.

Parameters

- **id** (*int*) – Id of the source package.
- **name** (*str*) – Name of the source package.
- **description** (*str*) – Description of the schedule.
- **is_deprecated** (*bool*) – Whether or not the source package is deprecated.
- **package_name** (*str*) – The name of the package containing the model.py file.
- **is_uploaded** (*bool*) – Whether or not the source package has been uploaded
- **available_operations** (*List[str]*) – The available operations on this source package. Can be any of [PREDICT, TRAIN].
- **created_time** (*int*) – Created time in UNIX.
- **runtime_version** (*str*) – The runtime version this source package should be deployed with. Can be any of ["0.1"]
- **metadata** (*Dict*) – User-defined metadata about the source package.
- **cognite_client** (CogniteClient) – An optional CogniteClient to associate with this data class.

class cognite.client.data_classes.model_hosting.source_packages.**SourcePackageList**(*resources: List[Any], cognite_client=*

Bases: *cognite.client.data_classes._base.CogniteResourceList*

2.5.2 Relationships

Warning: The relationships API is experimental and subject to breaking changes. It should not be used in production code.

Retrieve a relationship by id

RelationshipsAPI.**retrieve**(*external_id: str*) → Optional[cognite.client.data_classes.relationships.Relationship]

Retrieve a single relationship by external id.

Parameters **external_id** (*str*) – External ID

Returns Requested relationship or None if it does not exist.

Return type Optional[*Relationship*]

Examples

Get relationship by external id:

```
>>> from cognite.client.experimental import CogniteClient
>>> c = CogniteClient()
>>> res = c.relationships.retrieve(external_id="1")
```

Retrieve multiple relationships by id

RelationshipsAPI.**retrieve_multiple**(*external_ids: List[str]*) → cognite.client.data_classes.relationships.RelationshipList

Retrieve multiple relationships by external id.

Parameters **external_ids** (*List[str]*) – External IDs

Returns The requested relationships.

Return type *RelationshipList*

Examples

Get relationships by external id:

```
>>> from cognite.client.experimental import CogniteClient
>>> c = CogniteClient()
>>> res = c.relationships.retrieve_multiple(external_ids=["abc", "def"])
```

List relationships

RelationshipsAPI.**list**(*source_resource: str = None, source_resource_id: str = None, sources: List[Dict[str, Any]] = None, target_resource: str = None, target_resource_id: str = None, targets: List[Dict[str, Any]] = None, start_time: Dict[str, Any] = None, end_time: Dict[str, Any] = None, confidence: Dict[str, Any] = None, last_updated_time: Dict[str, Any] = None, created_time: Dict[str, Any] = None, data_set: Union[str, List[str], None] = None, relationship_type: Union[str, List[str], None] = None, limit: int = 25*) → cognite.client.data_classes.relationships.RelationshipList

List relationships

Parameters

- **source_resource** (*str*) – Resource type of the source node.
- **source_resource_id** (*str*) – Resource ID of the source node.
- **sources** (*List[Dict[str, Any]]*) – List of multiple sources in the format *[{"resourceId":externalId,"resource":"Asset"},...]*
- **target_resource** (*str*) – Resource type of the target node.
- **target_resource_id** (*str*) – Resource ID of the target node.
- **targets** (*List[Dict[str, Any]]*) – List of multiple targets in the format *[{"resourceId":externalId,"resource":"Asset"},...]*
- **start_time** (*Dict[str, Any]*) – Range to filter the field for. (inclusive)
- **end_time** (*Dict[str, Any]*) – Range to filter the field for. (inclusive)
- **confidence** (*Dict[str, Any]*) – Range to filter the field for. (inclusive)
- **last_updated_time** (*Dict[str, Any]*) – Range to filter the field for. (inclusive)
- **created_time** (*Dict[str, Any]*) – Range to filter the field for. (inclusive)
- **data_set** (*Union[str,List[str]*) – Filter on any of a given list of dataSets.

- **relationship_type** (*Union[str,List[str]]*) – Filter on any of a given list o relationship types.
- **limit** (*int, optional*) – Maximum number of relationships to return. Defaults to 100. Set to -1, float("inf") or None to return all items.

Returns List of requested relationships

Return type *RelationshipList*

Examples

List relationships:

```
>>> from cognite.client.experimental import CogniteClient
>>> c = CogniteClient()
>>> relationship_list = c.relationships.list(limit=5)
```

Iterate over relationships:

```
>>> from cognite.client.experimental import CogniteClient
>>> c = CogniteClient()
>>> for relationship in c.relationships:
...     relationship # do something with the relationship
```

Iterate over chunks of relationships to reduce memory load:

```
>>> from cognite.client.experimental import CogniteClient
>>> c = CogniteClient()
>>> for relationship_list in c.relationships(chunk_size=2500):
...     relationship_list # do something with the relationships
```

Create a relationship

RelationshipsAPI.**create**(*relationship: Union[cognite.client.data_classes.relationships.Relationship, List[cognite.client.data_classes.relationships.Relationship]]*) → Union[cognite.client.data_classes.relationships.Relationship, cognite.client.data_classes.relationships.RelationshipList]

Create one or more relationships.

Parameters **relationship** (*Union[Relationship, List[Relationship]]*) – Relationship or list of relationships to create. Note: the source and target field in the Relationship(s) can be of the form shown below, or objects of type Asset, TimeSeries, FileMetadata, Event, Sequence

Returns Created relationship(s)

Return type Union[*Relationship*, *RelationshipList*]

Examples

Create a new relationship specifying object type and external id for source and target:

```
>>> from cognite.client.experimental import CogniteClient
>>> from cognite.client.data_classes import Relationship
>>> c = CogniteClient()
>>> rel = Relationship(external_id="rel",source={"resource":"TimeSeries",
↪"resourceId": "ts"},target={"resource":"Asset", "resourceId": "a"},relationship_
↪type="belongsTo",confidence=0.9,data_set="ds_name")
>>> res = c.relationships.create(rel)
```

Create a new relationship using objects directly as source and target:

```
>>> from cognite.client.experimental import CogniteClient
>>> from cognite.client.data_classes import Relationship
>>> c = CogniteClient()
>>> assets = c.assets.retrieve_multiple(id=[1,2,3])
>>> flowrel1 = Relationship(external_id="flow_1",source=assets[0],
↪target=assets[1] ,relationship_type="flowsTo",confidence=0.1,data_set="ds_flow")
>>> flowrel2 = Relationship(external_id="flow_2",source=assets[1],
↪target=assets[2] ,relationship_type="flowsTo",confidence=0.1,data_set="ds_flow")
>>> res = c.relationships.create([flowrel1,flowrel2])
```

Delete relationships

`RelationshipsAPI.`**`delete`**(*external_id: Union[str, List[str]]*) → None

Delete one or more relationships

Parameters **`external_id`** (*`Union[str, List[str]]`*) – External ID or list of external ids

Returns None

Examples

Delete relationships by external id:

```
>>> from cognite.client.experimental import CogniteClient
>>> c = CogniteClient()
>>> c.relationships.delete(external_id=["a","b"])
```

Data classes

class cognite.client.data_classes.relationships.**Relationship**(*source: Dict[str, Any] = None*, *target: Dict[str, Any] = None*, *start_time: int = None*, *end_time: int = None*, *confidence: float = None*, *data_set: str = None*, *external_id: str = None*, *relationship_type: str = None*, *created_time: int = None*, *last_updated_time: int = None*, *cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResource*

Representation of a relationship in CDF, consists of a source and a target and some additional parameters.

Parameters

- **source** (*Dict[str, Any]*) – Reference by external id to the source of the relationship. Since it is a reference by external id, the targeted resource may or may not exist in CDF. If resource is *threeD* or *threeDRevision* the *resourceId* is a set of internal ids concatenated by a colons. Otherwise, the resourceId follows the formatting rules as described in *resourceId*. If resource id of type *threeD*, the externalId must follow the pattern <*nodeId*>:<*modelId*>:<*revisionId*>. If resource id of type *threeDRevision*, the externalId must follow the pattern <*revisionId*>:<*modelId*>. The values <*nodeId*>, <*modelId*> and <*revisionId*> are the corresponding internal ids to identify the referenced resource uniquely.
- **target** (*Dict[str, Any]*) – Reference by external id to the target of the relationship. Since it is a reference by external id, the targeted resource may or may not exist in CDF. If resource is *threeD* or *threeDRevision* the *resourceId* is a set of internal ids concatenated by a colons. Otherwise, the resourceId follows the formatting rules as described in *resourceId*. If resource id of type *threeD*, the externalId must follow the pattern <*nodeId*>:<*modelId*>:<*revisionId*>. If resource id of type *threeDRevision*, the externalId must follow the pattern <*revisionId*>:<*modelId*>. The values <*nodeId*>, <*modelId*> and <*revisionId*> are the corresponding internal ids to identify the referenced resource uniquely.
- **start_time** (*int*) – Time, in milliseconds since Jan. 1, 1970, when relationship became active. If there is no startTime, relationship is active from the beginning of time until endTime.
- **end_time** (*int*) – Time, in milliseconds since Jan. 1, 1970, when relationship became inactive. If there is no endTime, relationship is active from startTime until the present or any point in the future. If endTime and startTime are set, then endTime must be strictly greater than startTime
- **confidence** (*float*) – Confidence value of the existence of this relationship. Humans should enter 1.0 usually, generated relationships should provide a realistic score on the likelihood of the existence of the relationship. Generated relationships should never have the a confidence score of 1.0.

- **data_set** (*str*) – String describing the source system storing or generating the relationship.
- **external_id** (*str*) – Disallowing leading and trailing whitespaces. Case sensitive. The external Id must be unique within the project.
- **relationship_type** (*str*) – Type of the relationship in order to distinguish between different relationships. In general relationship types should reflect references as they are expressed in natural sentences. E.g. a flow through a pipe can be naturally represented by a *flowsTo*-relationship. On the other hand an alternative asset hierarchy can be represented with the *isParentOf*-relationship. The *implements*-relationship is intended to reflect references between a functional asset hierarchy and its implementation.
- **created_time** (*int*) – Time, in milliseconds since Jan. 1, 1970, when this relationship was created in CDF.
- **last_updated_time** (*int*) – Time, in milliseconds since Jan. 1, 1970, when this relationship was last updated in CDF.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.relationships.**RelationshipFilter**(*sources: List[Dict[str, Any]] = None*, *targets: List[Dict[str, Any]] = None*, *relationship_types: List[str] = None*, *data_sets: List[str] = None*, *start_time: Dict[str, Any] = None*, *end_time: Dict[str, Any] = None*, *confidence: Dict[str, Any] = None*, *last_updated_time: Dict[str, Any] = None*, *created_time: Dict[str, Any] = None*, *active_at_time: int = None*, *source_resource: str = None*, *source_resource_id: str = None*, *target_resource: str = None*, *target_resource_id: str = None*, *data_set: str = None*, *relationship_type: str = None*, *cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteFilter*

Filter on relationships with exact match. Multiple filter elments in one property, e.g. *dataSets: ["a", "b"]*, will return all relationships where the dataSet field is either *a* or *b*. Filters in multiple properties will return the relationships that match all criteria. Filters on a *resourceId* without a *resource* (type) in sources and targets will return relationships that match the resourceId and match any resource type.

Parameters

- **sources** (*List[Dict[str, Any]]*) – Include relationships that have any of these values in their *source* field
- **targets** (*List[Dict[str, Any]]*) – Include relationships that have any of these values in their *target* field
- **relationship_types** (*List[str]*) – Include relationships that have any of these values in their *relationshipType* field
- **data_sets** (*List[str]*) – Include relationships that have any of these values in their *dataSet* field
- **start_time** (*Dict[str, Any]*) – Range to filter the field for. (inclusive)
- **end_time** (*Dict[str, Any]*) – Range to filter the field for. (inclusive)
- **confidence** (*Dict[str, Any]*) – Range to filter the field for. (inclusive)
- **last_updated_time** (*Dict[str, Any]*) – Range to filter the field for. (inclusive)
- **created_time** (*Dict[str, Any]*) – Range to filter the field for. (inclusive)
- **active_at_time** (*int*) – Limits results to those active at this time, i.e. activeAtTime falls between startTime and endTime startTime is treated as inclusive (if activeAtTime is equal to startTime then the relationship will be included). endTime is treated as exclusive (if activeTime is equal to endTime then the relationsip will NOT be included). If a relationship has neither startTime nor endTime, the relationship is active at all times
- **source_resource** (*str*) – Resource type of the source node.
- **source_resource_id** (*str*) – Resource ID of the source node.
- **target_resource** (*str*) – Resource type of the target node.
- **target_resource_id** (*str*) – Resource ID of the target node.
- **data_set** (*str*) – String describing the source system storing or generating the relationship.
- **relationship_type** (*str*) – Type of the relationship in order to distinguish between different relationships. In general relationship types should reflect references as they are expressed in natural sentences.
- **cognite_client** (CogniteClient) – The client to associate with this object.

class cognite.client.data_classes.relationships.**RelationshipList**(*resources: List[Any], cognite_client=None*)

Bases: *cognite.client.data_classes._base.CogniteResourceList*

2.5.3 Synthetic time series

Warning: The synthetic time series API is experimental and subject to breaking changes. It should not be used in production code.

Calculate the result of a function on time series

SyntheticDatapointsAPI.**retrieve**(*expression: Union[str, sympy.Expr], start: Union[int, str, datetime.datetime], end: Union[int, str, datetime.datetime], limit: int = None, variables: Dict[str, Union[str, cognite.client.data_classes.time_series.TimeSeries]] = None, aggregate: str = None, granularity: str = None*) → cognite.client.data_classes.datapoints.Datapoints

Calculate the result of a function on time series.

Parameters

- **expression** (*Union[str,sympy.Expr]*) – Function to be calculated. Supports both strings and sympy expressions. Strings can have either the API *ts{}* syntax, or contain variable names to be replaced using the *variables* parameter.
- **start** (*Union[int, str, datetime]*) – Inclusive start.
- **end** (*Union[int, str, datetime]*) – Exclusive end.
- **limit** (*int*) – Number of datapoints to retrieve.
- **variables** (*Dict[str,Union[str,TimeSeries]]*) – An optional map of symbol replacements.
- **aggregate** (*str*) – use this aggregate when replacing entries from *variables*, does not affect time series given in the *ts{}* syntax.
- **granularity** (*str*) – use this granularity with the aggregate.

Returns A Datapoints object containing the calculated data.

Return type *Datapoints*

Examples

Request a synthetic time series query with direct syntax

```
>>> from cognite.client.experimental import CogniteClient
>>> c = CogniteClient()
>>> dps = c.datapoints.synthetic.retrieve(expression="TS{id:123} + TS{externalId:
↪'abc'}", start="2w-ago", end="now")
```

Use variables to re-use an expression:

```
>>> from cognite.client.experimental import CogniteClient
>>> c = CogniteClient()
>>> vars = {"A": "my_ts_external_id", "B": client.time_series.retrieve(id=1)}
>>> dps = c.datapoints.synthetic.retrieve(expression="A+B", start="2w-ago", end=
↪"now", variables=vars)
```

Use sympy to build complex expressions:

```
>>> from cognite.client.experimental import CogniteClient
>>> c = CogniteClient()
>>> from sympy import symbols, cos, pi
>>> a = sympy.symbols('a')
>>> dps = c.datapoints.synthetic.retrieve(pi * cos(a), start="2w-ago", end="now",␣
↪variables={"a": "my_ts_external_id"},aggregate='interpolation',granularity='1m')
```

CHAPTER 3

Examples

For a collection of scripts and Jupyter Notebooks that explain how to perform various tasks in Cognite Data Fusion (CDF) using Python, see the GitHub repository here.

Python Module Index

C

Index

D

E

F

G

I

L

S

T

U

V