
cognite-sdk-python Documentation

Release 5.12.0

Cognite

May 22, 2023

Contents

1	Installation	3
2	Contents	5
2.1	Quickstart	5
2.2	Settings	8
2.3	Extensions and optional dependencies	9
2.4	API	10
	Python Module Index	295
	Index	297

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).

- *Installation*
- *Contents*

CHAPTER 1

Installation

To install this package:

```
pip install cognite-sdk
```

To upgrade the version of this package:

```
pip install cognite-sdk --upgrade
```

```
pip install "cognite-sdk[pandas, geo]"
```


2.1 Quickstart

2.1.1 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 a user-defined string intended to give the client a unique identifier. You can provide the `client_name` by passing it directly to the `ClientConfig` constructor.

The Cognitive API uses OpenID Connect (OIDC) to authenticate. Use one of the credential providers such as `OAuthClientCredentials` to authenticate:

Note: The following example sets a global client configuration which will be used if no config is explicitly passed to `CognitoClient`. All examples in this documentation assume that such a global configuration has been set.

```
from cognite.client import CognitoClient, ClientConfig, global_config
from cognite.client.credentials import OAuthClientCredentials

# This value will depend on the cluster your CDF project runs on
cluster = "api"
base_url = f"https://{cluster}.cognitedata.com"
tenant_id = "my-tenant-id"
client_id = "my-client-id"
# client secret should not be stored in-code, so we load it from an environment_
↪variable
client_secret = os.environ["MY_CLIENT_SECRET"]
creds = OAuthClientCredentials(
    token_url=f"https://login.microsoftonline.com/{tenant_id}/oauth2/v2.0/token",
    client_id=client_id,
    client_secret=client_secret,
    scopes=[f"{base_url}/.default"]
```

(continues on next page)

(continued from previous page)

```
)  
  
cnf = ClientConfig(  
    client_name="my-special-client",  
    base_url=base_url,  
    project="my-project",  
    credentials=creds  
)  
  
global_config.default_client_config = cnf  
c = CogniteClient()
```

Examples for all OAuth credential providers can be found in the *Credential Providers* section.

You can also make your own credential provider:

```
from cognite.client import CogniteClient, ClientConfig  
from cognite.client.credentials import Token  
  
def token_provider():  
    ...  
  
cnf = ClientConfig(  
    client_name="my-special-client",  
    base_url="https://<cluster>.cognitedata.com",  
    project="my-project",  
    credentials=Token(token_provider)  
)  
c = CogniteClient(cnf)
```

2.1.2 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()
```

2.1.3 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. Note that all assets must have a name (a non-empty string).

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 (or parent external ID):

```

from cognite.client import CogniteClient
from cognite.client.data_classes import Asset

c = CogniteClient()
my_asset = Asset(name="my first child 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 created in the correct order, you can use the `create_hierarchy()` function, which takes care of the topological sorting for you, 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 creating the assets fails:

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

```

from cognite.client.exceptions import CogniteAPIError
try:
    c.assets.create_hierarchy([root, child, descendant])
except CogniteAPIError as err:
    created = err.successful
    maybe_created = err.unknown
    not_created = err.failed

```

Prior to creating the Assets, it might be useful to do some validation on the assets you have. To do this without potentially sending API requests, import and use `AssetHierarchy`:

```

from cognite.client.data_classes import AssetHierarchy
hierarchy = AssetHierarchy(assets)
# Get a report written to the terminal listing any issues:
hierarchy.validate_and_report()
# If there are issues, you may inspect them directly:
if not hierarchy.is_valid():
    hierarchy.orphans
    hierarchy.invalid
    hierarchy.unsure_parents
    hierarchy.duplicates
    hierarchy.cycles # Requires no other basic issues

```

Note that validation will run automatically for you when calling `create_hierarchy()`. You may choose to catch `CogniteAssetHierarchyError` and inspect any raised issues:

```
from cognite.client.exceptions import CogniteAssetHierarchyError
try:
    c.assets.create_hierarchy(assets)
except CogniteAssetHierarchyError as err:
    # You may inspect the following attributes:
    err.orphans
    err.invalid
    err.unsure_parents
    err.duplicates
    err.cycles # Requires no other basic issues
```

2.1.4 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"
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 global configuration

You can set global configuration options like this:

```
from cognite.client import global_config, ClientConfig
from cognite.client.credentials import APIKey
global_config.default_client_config = ClientConfig(
    client_name="my-client", project="myproj", credentials=APIKey("verysecret")
)
```

(continues on next page)

(continued from previous page)

```
global_config.disable_pypi_version_check = True
global_config.disable_gzip = False
global_config.disable_ssl = False
global_config.max_retries = 10
global_config.max_retry_backoff = 10
global_config.max_connection_pool_size = 10
global_config.status_forcelist = {429, 502, 503}
```

These must be set prior to instantiating a `CogniteClient` in order for them to take effect.

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 `max_workers` config option.

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 `max_connection_pool_size` config option.

2.3 Extensions and optional dependencies

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 How to install extra dependencies

If your application requires the functionality from e.g. the `pandas`, `sympy`, or `geopandas` dependencies, you should install the SDK along with its optional dependencies. The available extras are:

- `numpy`: `numpy`
- `pandas`: `pandas`
- `geo`: `geopandas`, `shapely`
- `sympy`: `sympy`
- `functions`: `pip`
- `all` (will install dependencies for all the above)

These can be installed with the following command:

```
pip
```

```
$ pip install "cognite-sdk[pandas, geo]"
```

```
poetry
```

```
$ poetry add cognite-sdk -E pandas -E geo
```

2.4 API

2.4.1 CogniteClient

class `cognite.client.CogniteClient` (*config*: `Optional[cognite.client.config.ClientConfig]` = `None`)

Main entrypoint into Cognite Python SDK.

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

Parameters `config` (`ClientConfig`) – The configuration for this client.

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

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

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

delete (*url*: `str`, *params*: `Optional[Dict[str, Any]]` = `None`, *headers*: `Optional[Dict[str, Any]]` = `None`) → `requests.models.Response`
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`

class `cognite.client.config.ClientConfig` (*client_name*: `str`, *project*: `str`, *credentials*: `cognite.client.credentials.CredentialProvider`, *api_subversion*: `Optional[str]` = `None`, *base_url*: `Optional[str]` = `None`, *max_workers*: `Optional[int]` = `None`, *headers*: `Optional[Dict[str, str]]` = `None`, *timeout*: `Optional[int]` = `None`, *file_transfer_timeout*: `Optional[int]` = `None`, *debug*: `bool` = `False`)

Configuration object for the client

Parameters

- **client_name** (`str`) – A user-defined name for the client. Used to identify number of unique applications/scripts running on top of CDF.
- **project** (`str`) – CDF Project name.

- **credentials** (*CredentialProvider*) – Credentials. e.g. APIKey, Token, Client-Credentials.
- **api_subversion** (*str*) – API subversion
- **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.
- **file_transfer_timeout** (*int*) – Timeout on file upload/download requests. Defaults to 600 seconds.
- **debug** (*bool*) – Configures logger to log extra request details to stderr.

class `cognite.client.config.GlobalConfig`

Global configuration object

default_client_config

A default instance of a client configuration. This will be used by the CogniteClient constructor if no config is passed directly. Defaults to None.

Type Optional[*ClientConfig*]

disable_gzip

Whether or not to disable gzipping of json bodies. Defaults to False.

Type bool

disable_pypi_version_check

Whether or not to check for newer SDK versions when instantiating a new client. Defaults to False.

Type bool

status_forcelist

HTTP status codes to retry. Defaults to {429, 502, 503, 504}

Type Set[int]

max_retries

Max number of retries on a given http request. Defaults to 10.

Type int

max_retries_connect

Max number of retries on connection errors. Defaults to 3.

Type int

max_retry_backoff

Retry strategy employs exponential backoff. This parameter sets a max on the amount of backoff after any request failure. Defaults to 30.

Type int

max_connection_pool_size

The maximum number of connections which will be kept in the SDKs connection pool. Defaults to 50.

Type int

disable_ssl

Whether or not to disable SSL. Defaults to False

Type bool

proxies

Dictionary mapping from protocol to url. e.g. {"https": "http://10.10.1.10:1080"}

Type Dict[str, str]

2.4.2 Authentication

Credential Providers

class `cognite.client.credentials.Token` (*token: Union[str, Callable[[], str]]*)

Token credential provider

Parameters `token` (*Union[str, Callable[[], str]*) – A token or a token factory.

Examples

```
>>> from cognite.client.credentials import Token
>>> token_provider = Token("my secret token")
>>> token_factory_provider = Token(lambda: "my secret token")
```

Note: If you pass in a callable, we will expect that you supplied a function that may do a token refresh under the hood, so it will be called while holding a thread lock (`threading.Lock()`).

class `cognite.client.credentials.OAuthClientCredentials` (*token_url: str, client_id: str, client_secret: str, scopes: List[str], **token_custom_args*)

OAuth credential provider for the “Client Credentials” flow.

Parameters

- **token_url** (*str*) – OAuth token url
- **client_id** (*str*) – Your application’s client id.
- **client_secret** (*str*) – Your application’s client secret
- **scopes** (*List[str]*) – A list of scopes.
- ****token_custom_args** (*Any*) – Optional additional arguments to pass as query parameters to the token fetch request.

Examples

```
>>> from cognite.client.credentials import OAuthClientCredentials
>>> import os
>>> oauth_provider = OAuthClientCredentials(
...     token_url="https://login.microsoftonline.com/xyz/oauth2/v2.0/token",
...     client_id="abcd",
...     client_secret=os.environ["OAUTH_CLIENT_SECRET"],
...     scopes=["https://greenfield.cognitedata.com/.default"],
...     # Any additional IDP-specific token args. e.g.
```

(continues on next page)

(continued from previous page)

```
...     audience="some-audience"
... )
```

```
class cognite.client.credentials.OAuthInteractive(authority_url: str, client_id: str,
                                                scopes: List[str], redirect_port: int
                                                = 53000, token_cache_path: Op-
                                                tional[pathlib.Path] = None)
```

OAuth credential provider for an interactive login flow.

Make sure you have <http://localhost:port> in Redirect URI in App Registration as type “Mobile and desktop applications”.

Parameters

- **authority_url** (*str*) – OAuth authority url
- **client_id** (*str*) – Your application’s client id.
- **scopes** (*List[str]*) – A list of scopes.
- **redirect_port** (*List[str]*) – Redirect port defaults to 53000.
- **token_cache_path** (*Path*) – Location to store token cache, defaults to os temp directory/cognitetokencache.{client_id}.bin.

Examples

```
>>> from cognite.client.credentials import OAuthInteractive
>>> oauth_provider = OAuthInteractive(
...     authority_url="https://login.microsoftonline.com/xyz",
...     client_id="abcd",
...     scopes=["https://greenfield.cognitedata.com/.default"],
... )
```

```
class cognite.client.credentials.OAuthDeviceCode(authority_url: str, client_id: str,
                                                scopes: List[str], token_cache_path:
                                                Optional[pathlib.Path] = None)
```

OAuth credential provider for the device code login flow.

Parameters

- **authority_url** (*str*) – OAuth authority url
- **client_id** (*str*) – Your application’s client id.
- **scopes** (*List[str]*) – A list of scopes.
- **token_cache_path** (*Path*) – Location to store token cache, defaults to os temp directory/cognitetokencache.{client_id}.bin.

Examples

```
>>> from cognite.client.credentials import OAuthDeviceCode
>>> oauth_provider = OAuthDeviceCode(
...     authority_url="https://login.microsoftonline.com/xyz",
...     client_id="abcd",
...     scopes=["https://greenfield.cognitedata.com/.default"],
... )
```

```
class cognite.client.credentials.OAuthClientCertificate (authority_url: str;
                                                         client_id: str;
                                                         cert_thumbprint: str;
                                                         certificate: str; scopes:
                                                         List[str])
```

OAuth credential provider for authenticating with a client certificate.

Parameters

- **authority_url** (*str*) – OAuth authority url
- **client_id** (*str*) – Your application’s client id.
- **cert_thumbprint** (*str*) – Your certificate’s thumbprint. You get it when you upload your certificate to Azure AD.
- **certificate** (*str*) – Your private certificate, typically read from a .pem file
- **scopes** (*List[str]*) – A list of scopes.

Examples

```
>>> from cognite.client.credentials import OAuthClientCertificate
>>> from pathlib import Path
>>> oauth_provider = OAuthClientCertificate(
...     authority_url="https://login.microsoftonline.com/xyz",
...     client_id="abcd",
...     cert_thumbprint="XYZ123",
...     certificate=Path("certificate.pem").read_text(),
...     scopes=["https://greenfield.cognitedata.com/.default"],
... )
```

Get login status

`LoginAPI.status()` → `cognite.client.data_classes.login.LoginStatus`

Check login status

Note: This endpoint is only applicable if the client has been authenticated with an api key. If not, use `client.iam.token.inspect` instead

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 () → pandas.DataFrame

2.4.3 Assets

Retrieve an asset by id

```
AssetsAPI.retrieve (id: Optional[int] = None, external_id: Optional[str] = None) → Op-
tional[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[Sequence[int]] = None*, *external_ids*: *Optional[Sequence[str]] = None*, *ignore_unknown_ids*: *bool = False*) → `cognite.client.data_classes.assets.AssetList`

Retrieve multiple assets by id.

Parameters

- **ids** (*Sequence[int]*, *optional*) – IDs
- **external_ids** (*Sequence[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*: *Optional[int] = None*, *external_id*: *Optional[str] = None*, *depth*: *Optional[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 or empty `AssetList` if asset does not exist.

Return type *AssetList*

List assets

`AssetsAPI.list` (*name*: `Optional[str] = None`, *parent_ids*: `Optional[Sequence[int]] = None`, *parent_external_ids*: `Optional[Sequence[str]] = None`, *asset_subtree_ids*: `Union[int, Sequence[int], None] = None`, *asset_subtree_external_ids*: `Union[str, Sequence[str], None] = None`, *data_set_ids*: `Union[int, Sequence[int], None] = None`, *data_set_external_ids*: `Union[str, Sequence[str], None] = None`, *labels*: `Optional[cognite.client.data_classes.labels.LabelFilter] = None`, *geo_location*: `Optional[cognite.client.data_classes.shared.GeoLocationFilter] = None`, *metadata*: `Optional[Dict[str, str]] = None`, *source*: `Optional[str] = None`, *created_time*: `Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None`, *last_updated_time*: `Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None`, *root*: `Optional[bool] = None`, *external_id_prefix*: `Optional[str] = None`, *aggregated_properties*: `Optional[Sequence[str]] = None`, *partitions*: `Optional[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** (*Sequence[int]*) – Return only the direct descendants of the specified assets.
- **parent_external_ids** (*Sequence[str]*) – Return only the direct descendants of the specified assets.
- **asset_subtree_ids** (*Union[int, Sequence[int]]*) – Asset subtree id or list of asset subtree ids to filter on.
- **asset_subtree_external_ids** (*Union[str, Sequence[str]]*) – Asset subtree external id or list of asset subtree external ids to filter on.
- **data_set_ids** (*Union[int, Sequence[int]]*) – Return only assets in the specified data set(s) with this id / these ids.
- **data_set_external_ids** (*Union[str, Sequence[str]]*) – Return only assets in the specified data set(s) with this external id / these external ids.
- **labels** (*LabelFilter*) – Return only the assets matching the specified label filter.
- **geo_location** (*GeoLocationFilter*) – Only include files matching the specified geographic relation.
- **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** (*Sequence[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
```

Filter assets based on labels:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import LabelFilter
>>> c = CogniteClient()
>>> my_label_filter = LabelFilter(contains_all=["PUMP", "VERIFIED"])
>>> asset_list = c.assets.list(labels=my_label_filter)
```

Aggregate assets

`AssetsAPI.aggregate` (*filter: Union[cognite.client.data_classes.assets.AssetFilter, dict, None] = 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"
↳ ""})
```

Aggregate asset metadata keys

`AssetsAPI.aggregate_metadata_keys` (*filter*: `Union[cognite.client.data_classes.assets.AssetFilter, dict, None]` = `None`) → `Sequence[cognite.client.data_classes.shared.AggregateBucketResult]`

Aggregate assets

Note: In the case of text fields, the values are aggregated in a case-insensitive manner

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

Returns List of asset aggregates

Return type `Sequence[AggregateBucketResult]`

Examples

Aggregate assets:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> aggregate_by_prefix = c.assets.aggregate_metadata_keys(filter={"external_id_
↳ prefix": "prefix"})
```

Aggregate asset metadata values

`AssetsAPI.aggregate_metadata_values` (*keys*: `Sequence[str]`, *filter*: `Union[cognite.client.data_classes.assets.AssetFilter, dict, None]` = `None`) → `Sequence[cognite.client.data_classes.shared.AggregateBucketResult]`

Aggregate assets

Note: In the case of text fields, the values are aggregated in a case-insensitive manner

Parameters

- **filter** (`Union[AssetFilter, Dict]`) – Filter on assets filter with exact match
- **keys** (`Sequence[str]`) – Metadata key(s) to apply the aggregation on. Currently supports exactly one key per request.

Returns List of asset aggregates

Return type Sequence[AggregateBucketResult]

Examples

Aggregate assets:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> aggregate_by_prefix = c.assets.aggregate_metadata_values(
...     keys=["someKey"],
...     filter={"external_id_prefix": "prefix"}
... )
```

Search for assets

`AssetsAPI.search` (*name: Optional[str] = None, description: Optional[str] = None, query: Optional[str] = None, filter: Union[cognite.client.data_classes.assets.AssetFilter, Dict[KT, VT], None] = 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"})
```

Search for an asset with an attached label:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> my_label_filter = LabelFilter(contains_all=["PUMP"])
>>> res = c.assets.search(name="xyz", filter=AssetFilter(labels=my_label_filter))
```

Create assets

`AssetsAPI.create` (*asset*: `Union[cognite.client.data_classes.assets.Asset, Sequence[cognite.client.data_classes.assets.Asset]]`) → `Union[cognite.client.data_classes.assets.Asset, cognite.client.data_classes.assets.AssetId, cognite.client.data_classes.assets.AssetIdList]`

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 resolved into an internal ID and stored as *parentId*.

Parameters *asset* (`Union[Asset, Sequence[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 with label:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import Asset, Label
>>> c = CogniteClient()
>>> asset = Asset(name="my_pump", labels=[Label(external_id="PUMP")])
>>> res = c.assets.create(asset)
```

Create asset hierarchy

```
AssetsAPI.create_hierarchy (assets: Union[Sequence[cognite.client.data_classes.assets.Asset],  
cognite.client.data_classes.assets.AssetHierarchy], *, upsert: bool  
= False, upsert_mode: Literal[patch, replace] = 'patch') →  
cognite.client.data_classes.assets.AssetsList
```

Create an asset hierarchy with validation.

This helper function makes it easy to insert large asset hierarchies. It solves the problem of topological insertion order, i.e. a parent asset must exist before it can be referenced by any ‘children’ assets. You may pass any number of partial- or full hierarchies: there are no requirements on the number of root assets, so you may pass zero, one or many (same goes for the non-root assets).

Parameters

- **assets** (*Sequence[Asset] | AssetHierarchy*) – List of assets to create or an instance of AssetHierarchy.
- **upsert** (*bool*) – If used, already existing assets will be updated instead of an exception being raised. You may control how updates are applied with the ‘upsert_mode’ argument.
- **upsert_mode** (*"patch" | "replace"*) – Only applicable with upsert. Pass ‘patch’ to only update fields with non-null values (default), or ‘replace’ to do full updates (unset fields become null or empty).

Returns Created (and possibly updated) asset hierarchy

Return type *AssetsList*

Prior to insertion, this function will run validation on the given assets and raise an error if any of the following issues are found:

1. Any assets are invalid (category: *invalid*):
 - Missing external ID.
 - Missing a valid name.
 - Has an ID set.
2. Any asset duplicates exist (category: *duplicates*)
3. Any assets have an ambiguous parent link (category: *unsure_parents*)
4. Any group of assets form a cycle, e.g. A->B->A (category: *cycles*)

It is worth noting that validation is done “offline”, i.e. existing assets in CDF are not inspected. This means:

1. All assets linking a parent by ID are assumed valid
2. All orphan assets are assumed valid. “Orphan” means the parent is not part of the given assets (category: *orphans*)

Tip: The different categories specified above corresponds to the name of the attribute you might access on the raised error to get the collection of ‘bad’ assets falling in that group, e.g. `error.duplicates`.

Note: Updating `external_id` via `upsert` is not supported (and will not be supported). Use `AssetsAPI.update` instead.

Warning: The API does not natively support upsert, so the SDK has to simulate the behaviour at the cost of some insertion speed.

Be careful when moving assets to new parents via upsert: Please do so only by specifying `parent_external_id` (instead of `parent_id`) to avoid race conditions in insertion order (temporary cycles might form since we can only make changes to 1000 assets at the time).

Examples

Create an 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)
```

Create an asset hierarchy, but run update for existing assets:

```
>>> res = c.assets.create_hierarchy(assets, upsert=True, upsert_mode="patch")
```

Patch will only update the parameters you have defined on your assets. Note that specifically setting something to `None` is the same as not setting it. For metadata, this will extend your existing data, only overwriting when keys overlap. For labels the behaviour is mostly the same, existing are left untouched, and your new ones are simply added.

You may also pass `upsert_mode="replace"` to make sure the updated assets look identical to the ones you passed to the method. For both metadata and labels this will clear out all existing, before (potentially) adding the new ones.

If the hierarchy validation for some reason fail, you may inspect all the issues that were found by catching `CogniteAssetHierarchyError`:

```
>>> from cognite.client.exceptions import CogniteAssetHierarchyError
>>> try:
...     res = c.assets.create_hierarchy(assets)
... except CogniteAssetHierarchyError as err:
...     if err.invalid:
...         ... # do something
```

In addition to `invalid`, you may inspect `duplicates`, `unsure_parents`, `orphans` and `cycles`. Note that cycles are not available if any of the other basic issues exist, as the search for cyclical references requires a clean asset hierarchy to begin with.

You may also wrap the `create_hierarchy()` call in a try-except to get information if any of the assets fails to be created (assuming a valid hierarchy):

```
>>> from cognite.client.exceptions import CogniteAPIError
>>> try:
...     c.assets.create_hierarchy(assets)
... except CogniteAPIError as err:
...     created = err.successful
```

(continues on next page)

(continued from previous page)

```
... maybe_created = err.unknown
... not_created = err.failed
```

Here's a slightly longer explanation of the different groups:

- `err.successful`: Which assets were created (request yielded a 201)
- `err.unknown`: Which assets *may* have been created (request yielded 5xx)
- `err.failed`: Which assets were *not* created (request yielded 4xx, or was a descendant of an asset with unknown status)

The preferred way to create an asset hierarchy, is to run validation *prior to insertion*. You may do this by using the `AssetHierarchy` class. It will by default consider orphan assets to be problematic (but accepts the boolean parameter `ignore_orphans`), contrary to how `create_hierarchy` works (which accepts them in order to be backwards-compatible). It also provides helpful methods to create reports of any issues found, check out `validate_and_report`:

```
>>> from cognite.client.data_classes import AssetHierarchy
>>> from pathlib import Path
>>> hierarchy = AssetHierarchy(assets)
>>> if hierarchy.is_valid():
...     res = c.assets.create_hierarchy(hierarchy)
... else:
...     hierarchy.validate_and_report(output_file=Path("report.txt"))
```

Delete assets

`AssetsAPI.delete` (*id*: `Union[int, Sequence[int], None]` = `None`, *external_id*: `Union[str, Sequence[str], None]` = `None`, *recursive*: `bool` = `False`, *ignore_unknown_ids*: `bool` = `False`) → `None`
Delete one or more assets

Parameters

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

Update one or more assets Labels can be added, removed or replaced (set). Note that set operation deletes all the existing labels and adds the new specified labels.

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

Returns Updated asset(s)

Return type `Union[Asset, AssetList]`

Examples

Perform a partial update on an 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"})
>>> res1 = c.assets.update(my_update)
>>> # Remove an already set field like so
>>> another_update = AssetUpdate(id=1).description.set(None)
>>> res2 = c.assets.update(another_update)
```

Remove the metadata on an asset:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import AssetUpdate
>>> c = CogniteClient()
>>> my_update = AssetUpdate(id=1).metadata.add({"key": "value"})
>>> res1 = c.assets.update(my_update)
>>> another_update = AssetUpdate(id=1).metadata.set(None)
>>> # The same result can be achieved with:
>>> another_update2 = AssetUpdate(id=1).metadata.set({})
>>> res2 = c.assets.update(another_update)
```

Attach labels to an asset:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import AssetUpdate
>>> c = CogniteClient()
>>> my_update = AssetUpdate(id=1).labels.add(["PUMP", "VERIFIED"])
>>> res = c.assets.update(my_update)
```

Detach a single label from an asset:

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

(continues on next page)

(continued from previous page)

```
>>> my_update = AssetUpdate(id=1).labels.remove("PUMP")
>>> res = c.assets.update(my_update)
```

Replace all labels for an asset:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import AssetUpdate
>>> c = CogniteClient()
>>> my_update = AssetUpdate(id=1).labels.set("PUMP")
>>> res = c.assets.update(my_update)
```

Data classes

```
class cognite.client.data_classes.assets.AggregateResultItem (child_count:
                                                                Optional[int]
                                                                = None, depth:
                                                                Optional[int] =
                                                                None, path: Op-
                                                                tional[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, de-
                                                                scription: str = None, data_set_id:
                                                                int = None, metadata: Dict[str, str]
                                                                = None, source: str = None, la-
                                                                bel: List[Union[Label, str, LabelDef-
                                                                inition, dict]] = None, geo_location:
                                                                GeoLocation = None, id: int =
                                                                None, created_time: int = None,
                                                                last_updated_time: int = None, root_id:
                                                                int = None, aggregates: Union[Dict[str,
                                                                Any], AggregateResultItem] = None,
                                                                cognite_client: CogniteClient = 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.
- **labels** (*List[Label]*) – A list of the labels associated with this resource item.
- **geo_location** (*GeoLocation*) – The geographic metadata 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*

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]`

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: Optional[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: Sequence[str] = ('metadata', 'aggregates'), ignore: List[str] = None, camel_case: bool = False*) → pandas.DataFrame

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: Optional[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: Sequence[int] = None, parent_external_ids: Sequence[str] = None, asset_subtree_ids: Sequence[Dict[str, Any]] = None, data_set_ids: Sequence[Dict[str, Any]] = None, metadata: Dict[str, str] = None, source: str = None, created_time: Union[Dict[str, Any], TimestampRange] = None, last_updated_time: Union[Dict[str, Any], TimestampRange] = None, root: bool = None, external_id_prefix: str = None, labels: LabelFilter = None, geo_location: GeoLocationFilter = None, cognite_client: CogniteClient = None)

```

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

Filter on assets with strict matching.

Parameters

- **name** (*str*) – The name of the asset.
- **parent_ids** (*Sequence[int]*) – Return only the direct descendants of the specified assets.
- **parent_external_ids** (*Sequence[str]*) – Return only the direct descendants of the specified assets.
- **asset_subtree_ids** (*Sequence[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** (*Sequence[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.
- **labels** (*LabelFilter*) – Return only the resource matching the specified label constraints.

- **geo_location** (*GeoLocationFilter*) – Only include files matching the specified geographic relation.
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

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]

```
class cognite.client.data_classes.assets.AssetHierarchy (assets: Sequence[cognite.client.data_classes.assets.Asset], ignore_orphans: bool = False)
```

Bases: object

Class that verifies if a collection of Assets is valid, by validating its internal consistency. This is done “offline”, meaning CDF is *not* queried for the already existing assets. As a result, any asset providing a parent link by ID instead of external ID, are assumed valid.

Example usage:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import AssetHierarchy
>>> client = CogniteClient()
>>> hierarchy = AssetHierarchy(assets)
>>> # Get a report written to the terminal listing any issues:
>>> hierarchy.validate_and_report()
>>> if hierarchy.is_valid():
...     res = client.assets.create_hierarchy(hierarchy)
...     # If there are issues, you may inspect them directly:
... else:
...     hierarchy.orphans
...     hierarchy.invalid
...     hierarchy.unsure_parents
...     hierarchy.duplicates
...     hierarchy.cycles # Requires no other basic issues
```

There are other ways to generate the report than to write directly to screen. You may pass an *output_file* which can be either a Path object (writes are done in append-mode) or a file-like object supporting write (default is None which is just regular print):

```
>>> # Get a report written to file:
>>> from pathlib import Path
>>> report = Path("path/to/my_report.txt")
>>> hierarchy = AssetHierarchy(assets)
>>> hierarchy.validate_and_report(output_file=report)
```

```
>>> # Get a report as text "in memory":
>>> import io
>>> with io.StringIO() as file_like:
...     hierarchy.validate_and_report(output_file=file_like)
...     report = file_like.getvalue()
```

count_subtree (*mapping: Dict[Optional[str], List[cognite.client.data_classes.assets.Asset]]*) → Dict[str, int]

Returns a mapping from asset external ID to the size of its subtree (children and children of children etc.).

Parameters `mapping` (*Dict | None*) – The mapping returned by `groupby_parent_xid()`. If `None` is passed, will be recreated (slightly expensive).

Returns Lookup from external ID to descendant count.

Return type `Dict[str, int]`

`groupby_parent_xid()` → `Dict[Optional[str], List[cognite.client.data_classes.assets.Asset]]`

Returns a mapping from parent external ID to a list of its direct children.

Note: If the `AssetHierarchy` was initialized with `ignore_orphans=True`, all orphans assets, if any, are returned as part of the root assets in the mapping and can be accessed by `mapping[None]`. The same is true for all assets linking its parent by ID.

```
class cognite.client.data_classes.assets.AssetList (resources:      Collection[Any],
                                                  cognite_client: CogniteClient =
                                                  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: Optional[int] = None,
                                                       external_id: Optional[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 Labels

List labels

`LabelsAPI.list` (*name*: `Optional[str] = None`, *external_id_prefix*: `Optional[str] = None`, *data_set_ids*: `Union[int, Sequence[int], None] = None`, *data_set_external_ids*: `Union[str, Sequence[str], None] = None`, *limit*: `int = 25`) → `cognite.client.data_classes.labels.LabelDefinitionList`

List Labels

Parameters

- **name** (*str*) – returns the label definitions matching that name
- **data_set_ids** (`Union[int, Sequence[int]]`) – return only labels in the data sets with this id / these ids.
- **data_set_external_ids** (`Union[str, Sequence[str]]`) – return only labels in the data sets with this external id / these external ids.
- **external_id_prefix** (*str*) – filter label definitions with external ids starting with the prefix specified
- **limit** (*int, optional*) – Maximum number of label definitions to return.

Returns List of requested Labels

Return type `LabelDefinitionList`

Examples

List Labels and filter on name:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> label_list = c.labels.list(limit=5, name="Pump")
```

Iterate over label definitions:

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

Iterate over chunks of label definitions to reduce memory load:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for label_list in c.labels(chunk_size=2500):
...     label_list # do something with the type definitions
```

Create a label

`LabelsAPI.create` (*label*: `Union[cognite.client.data_classes.labels.LabelDefinition, Sequence[cognite.client.data_classes.labels.LabelDefinition]]`) → `Union[cognite.client.data_classes.labels.LabelDefinition, cognite.client.data_classes.labels.LabelDefinitionList]`

Create one or more label definitions.

Parameters **Label** (*Union[LabelDefinition, Sequence[LabelDefinition]]*) – label definition or a list of label definitions to create.

Returns Created label definition(s)

Return type *Union[LabelDefinition, LabelDefinitionList]*

Examples

Create new label definitions:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import LabelDefinition
>>> c = CogniteClient()
>>> labels = [LabelDefinition(external_id="ROTATING_EQUIPMENT", name="Rotating_
↳equipment"), LabelDefinition(external_id="PUMP", name="pump")]
>>> res = c.labels.create(labels)
```

Delete labels

`LabelsAPI.delete(external_id: Union[str, Sequence[str], None] = None) → None`

Delete one or more label definitions

Parameters **external_id** (*Union[str, Sequence[str]]*) – One or more label external ids

Returns None

Examples

Delete label definitions by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.labels.delete(external_id=["big_pump", "small_pump"])
```

Data classes

class `cognite.client.data_classes.labels.Label` (*external_id: Optional[str] = None, **kwargs*)

Bases: dict

A label assigned to a resource.

Parameters **external_id** (*str*) – The external id to the attached label.

class `cognite.client.data_classes.labels.LabelDefinition` (*external_id: str = None, name: str = None, description: str = None, created_time: int = None, data_set_id: int = None, cognite_client: CogniteClient = None*)

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

A label definition is a globally defined label that can later be attached to resources (e.g., assets). For example, can you define a “Pump” label definition and attach that label to your pump assets.

Parameters

- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.
- **name** (*str*) – Name of the label.
- **description** (*str*) – Description of the label.
- **created_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **data_set_id** (*int*) – The id of the dataset this label belongs to.
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

```
class cognite.client.data_classes.labels.LabelDefinitionFilter (name: str =  
None, external_id_prefix:  
str = None, data_set_ids:  
List[Dict[str, Any]] = None,  
cognite_client:  
CogniteClient = None)
```

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

Filter on labels definitions with strict matching.

Parameters

- **name** (*str*) – Returns the label definitions matching that name.
- **external_id_prefix** (*str*) – filter label definitions with external ids starting with the prefix specified
- **data_set_ids** (*List[Dict[str, Any]]*) – Only include labels that belong to these datasets.
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

```
class cognite.client.data_classes.labels.LabelDefinitionList (resources: Col-  
lection[Any],  
cognite_client:  
CogniteClient =  
None)
```

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

```
class cognite.client.data_classes.labels.LabelFilter (contains_any: List[str] =  
None, contains_all: List[str] =  
None, cognite_client: Cognite-  
Client = None)
```

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

Return only the resource matching the specified label constraints.

Parameters

- **contains_any** (*List[str]*) – The resource item contains at least one of the listed labels. The labels are defined by a list of external ids.

- **contains_all** (*List[str]*) – The resource item contains all the listed labels. The labels are defined by a list of external ids.
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

Examples

List only resources marked as a PUMP and VERIFIED:

```
>>> from cognite.client.data_classes import LabelFilter
>>> my_label_filter = LabelFilter(contains_all=["PUMP", "VERIFIED"])
```

List only resources marked as a PUMP or as a VALVE:

```
>>> from cognite.client.data_classes import LabelFilter
>>> my_label_filter = LabelFilter(contains_any=["PUMP", "VALVE"])
```

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]*

2.4.5 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[Sequence[int]] = None*, *external_ids*: *Optional[Sequence[str]] = None*, *ignore_unknown_ids*: *bool = False*) → `cognite.client.data_classes.events.EventList`

Retrieve multiple events by id.

Parameters

- **ids** (*Sequence[int]*, *optional*) – IDs
- **external_ids** (*Sequence[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] = None*, *end_time*: *Union[Dict[str, Any], cognite.client.data_classes.events.EndTimeFilter, None] = None*, *active_at_time*: *Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None*, *type*: *Optional[str] = None*, *subtype*: *Optional[str] = None*, *metadata*: *Optional[Dict[str, str]] = None*, *asset_ids*: *Optional[Sequence[int]] = None*, *asset_external_ids*: *Optional[Sequence[str]] = None*, *asset_subtree_ids*: *Union[int, Sequence[int], None] = None*, *asset_subtree_external_ids*: *Union[str, Sequence[str], None] = None*, *data_set_ids*: *Union[int, Sequence[int], None] = None*, *data_set_external_ids*: *Union[str, Sequence[str], None] = None*, *source*: *Optional[str] = None*, *created_time*: *Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None*, *last_updated_time*: *Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None*, *external_id_prefix*: *Optional[str] = None*, *sort*: *Optional[Sequence[str]] = None*, *partitions*: *Optional[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.
- **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.
- **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** (*Sequence[int]*) – Asset IDs of related equipments that this event relates to.
- **asset_external_ids** (*Sequence[str]*) – Asset External IDs of related equipment that this event relates to.
- **asset_subtree_ids** (*Union[int, Sequence[int]]*) – Asset subtree id or list of asset subtree ids to filter on.
- **asset_subtree_external_ids** (*Union[str, Sequence[str]]*) – Asset subtree external id or list of asset subtree external ids to filter on.
- **data_set_ids** (*Union[int, Sequence[int]]*) – Return only events in the specified data set(s) with this id / these ids.
- **data_set_external_ids** (*Sequence[str]*) – Return only events in the specified data set(s) with this external id / 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** (*Sequence[str]*) – Sort by array of selected fields. Ex: [“startTime:desc”]. Default sort order is asc when omitted. 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]` = `None`) → `List[cognite.client.data_classes.shared.AggregateResult]`

Aggregate events

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

Returns List of event aggregates

Return type `List[AggregateResult]`

Examples

Aggregate events:

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

`EventsAPI.aggregate_unique_values` (*filter*: `Union[cognite.client.data_classes.events.EventFilter, Dict[KT, VT], None]` = `None`, *fields*: `Optional[Sequence[str]]` = `None`) → `List[cognite.client.data_classes.shared.AggregateUniqueValuesResult]`

Aggregate unique values for events

Parameters

- **filter** (`Union[EventFilter, Dict]`) – Filter on events filter with exact match
- **fields** (`Sequence[str]`) – The field name(s) to apply the aggregation on. Currently limited to one field.

Returns List of event aggregates

Return type List[AggregateUniqueValuesResult]

Examples

Aggregate events:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> aggregate_subtype = c.events.aggregate_unique_values(filter={"type": "failure
↳"}, fields=["subtype"])
```

Search for events

EventsAPI.**search** (*description: Optional[str] = None, filter: Union[cognite.client.data_classes.events.EventFilter, Dict[KT, VT], None] = 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, Sequence[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, Sequence[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, Sequence[int], None]` = `None`, *external_id*: `Union[str, Sequence[str], None]` = `None`, *ignore_unknown_ids*: `bool` = `False`) → `None`

Delete one or more events

Parameters

- **id** (`Union[int, Sequence[int]]`) – Id or list of ids
- **external_id** (`Union[str, Sequence[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, Sequence[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, Sequence[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)
>>> 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: Optional[int] = None,
min: Optional[int] = None,
is_null: Optional[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: Sequence[int] = None,
source: str = None, id: int = None,
last_updated_time: int = None, cre-
ated_time: int = None, cognite_client:
CogniteClient = 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** (*Sequence[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.EventFilter (start_time: Union[Dict[str, Any], TimestampRange] = None, end_time: Union[Dict[str, Any], End-TimeFilter] = None, active_at_time: Union[Dict[str, Any], TimestampRange] = None, metadata: Dict[str, str] = None, asset_ids: Sequence[int] = None, asset_external_ids: Sequence[str] = None, asset_subtree_ids: Sequence[Dict[str, Any]] = None, data_set_ids: Sequence[Dict[str, Any]] = None, source: str = None, type: str = None, subtype: str = None, created_time: Union[Dict[str, Any], TimestampRange] = None, last_updated_time: Union[Dict[str, Any], TimestampRange] = None, external_id_prefix: str = None, cognite_client: CogniteClient = 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** (*Sequence[int]*) – Asset IDs of equipment that this event relates to.
- **asset_external_ids** (*Sequence[str]*) – Asset External IDs of equipment that this event relates to.
- **asset_subtree_ids** (*Sequence[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** (*Sequence[Dict[str, Any]]*) – Only include events that belong to these datasets.
- **source** (*str*) – The source of this event.
- **type** (*str*) – Type of the event, e.g ‘failure’.
- **subtype** (*str*) – SubType of the event, e.g ‘electrical’.
- **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:      Collection[Any],
                                                  cognite_client: CogniteClient =
                                                  None)
```

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

```
class cognite.client.data_classes.events.EventUpdate (id: Optional[int] = None,
                                                       external_id: Optional[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.

2.4.6 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[Sequence[int]] = None, *external_ids*: Optional[Sequence[str]] = None, *ignore_unknown_ids*: bool = False) → cognite.client.data_classes.data_sets.DataSetList

Retrieve multiple data sets by id.

Parameters

- **ids**(*Sequence[int]*, *optional*) – IDs
- **external_ids**(*Sequence[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: Optional[Dict[str, str]] = None, created_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None, last_updated_time: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None, external_id_prefix: Optional[str] = None, write_protected: Optional[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] = 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, Sequence[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, Sequence[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,
                                Sequence[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, Sequence[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: CogniteClient =
                                                    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. Limits: Maximum length of key is 128 bytes, value 10240 bytes, up to 256 key-value pairs, of total size at most 10240.
- **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:          Op-  
                                                         tional[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],          Timestamp-  
                                                         pRange]      = None,  
                                                         last_updated_time:  
                                                         Union[Dict[str, Any],  
                                                         TimestampRange]  
                                                         = None, external_id_prefix:  
                                                         str =  
                                                         None, write_protected:  
                                                         bool = None, cognite_client:  
                                                         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. Limits: Maximum length of key is 128 bytes, value 10240 bytes, up to 256 key-value pairs, of total size at most 10240.
- **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: Collection[Any], cognite_client: CogniteClient = None*)
 Bases: `cognite.client.data_classes._base.CogniteResourceList`

class `cognite.client.data_classes.data_sets.DataSetUpdate` (*id: Optional[int] = None, external_id: Optional[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.

2.4.7 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[Sequence[int]] = None`, *external_ids*: `Optional[Sequence[str]] = None`, *ignore_unknown_ids*: `bool = False`)
 → `cognite.client.data_classes.files.FileMetadataList`

Retrieve multiple file metadatas by id.

Parameters

- **ids** (`Sequence[int]`, *optional*) – IDs
- **external_ids** (`Sequence[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 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*: `Optional[str] = None`, *mime_type*: `Optional[str] = None`, *meta-data*: `Optional[Dict[str, str]] = None`, *asset_ids*: `Optional[Sequence[int]] = None`, *asset_external_ids*: `Optional[Sequence[str]] = None`, *asset_subtree_ids*: `Union[int, Sequence[int], None] = None`, *asset_subtree_external_ids*: `Union[str, Sequence[str], None] = None`, *data_set_ids*: `Union[int, Sequence[int], None] = None`, *data_set_external_ids*: `Union[str, Sequence[str], None] = None`, *labels*: `Optional[cognite.client.data_classes.labels.LabelFilter] = None`, *geo_location*: `Optional[cognite.client.data_classes.shared.GeoLocationFilter] = None`, *source*: `Optional[str] = None`, *created_time*: `Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None`, *last_updated_time*: `Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None`, *source_created_time*: `Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None`, *source_modified_time*: `Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None`, *uploaded_time*: `Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None`, *external_id_prefix*: `Optional[str] = None`, *directory_prefix*: `Optional[str] = None`, *uploaded*: `Optional[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** (*Sequence[int]*) – Only include files that reference these specific asset IDs.
- **asset_subtree_external_ids** (*Union[str, Sequence[str]]*) – Only include files that reference these specific asset external IDs.
- **asset_subtree_ids** (*Union[int, Sequence[int]]*) – Asset subtree id or list of asset subtree ids to filter on.
- **asset_subtree_external_ids** – Asset subtree external id or list of asset subtree external ids to filter on.
- **data_set_ids** (*Union[int, Sequence[int]]*) – Return only files in the specified data set(s) with this id / these ids.
- **data_set_external_ids** (*Sequence[str]*) – Return only files in the specified data set(s) with this external id / these external ids.
- **labels** (*LabelFilter*) – Return only the files matching the specified label filter(s).
- **geo_location** (*GeoLocationFilter*) – Only include files matching the specified geographic relation.
- **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.
- **directory_prefix** (*str*) – Filter by this (case-sensitive) prefix for the directory provided by the client.
- **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
```

Filter files based on labels:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import LabelFilter
>>> c = CogniteClient()
>>> my_label_filter = LabelFilter(contains_all=["WELL LOG", "VERIFIED"])
>>> file_list = c.files.list(labels=my_label_filter)
```

Filter files based on geoLocation:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import GeoLocationFilter, GeometryFilter
>>> c = CogniteClient()
>>> my_geo_location_filter = GeoLocationFilter(relation="intersects",
↳shape=GeometryFilter(type="Point", coordinates=[35,10]))
>>> file_list = c.files.list(geo_location=my_geo_location_filter)
```

Aggregate files metadata

`FilesAPI.aggregate` (*filter*: `Union[cognite.client.data_classes.files.FileMetadataFilter, Dict[KT, VT], None] = 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: Optional[str] = None, filter: Union[cognite.client.data_classes.files.FileMetadataFilter, dict, None] = None, limit: int = 100*) → `cog-`

`nite.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")
```

Search for an asset with an attached label:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> my_label_filter = LabelFilter(contains_all=["WELL LOG"])
>>> res = c.assets.search(name="xyz", filter=FileMetadataFilter(labels=my_label_
↪filter))
```

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: Optional[str] = None, name: Optional[str] = None, source: Optional[str] = None, mime_type: Optional[str] = None, metadata: Optional[Dict[str, str]] = None, directory: Optional[str] = None, asset_ids: Optional[Sequence[int]] = None, source_created_time: Optional[int] = None, source_modified_time: Optional[int] = None, data_set_id: Optional[int] = None, labels: Optional[Sequence[cognite.client.data_classes.labels.Label]] = None, geo_location: Optional[cognite.client.data_classes.shared.GeoLocation] = None, security_categories: Optional[Sequence[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.
- **directory** (*str*) – The directory to be associated with this file. Must be an absolute, unix-style path.
- **asset_ids** (*Sequence[int]*) – No description.
- **data_set_id** (*int*) – ID of the data set.

- **labels** (*Sequence[Label]*) – A list of the labels associated with this resource item.
- **geo_location** (*GeoLocation*) – The geographic metadata of the file.
- **security_categories** (*Sequence[int]*) – Security categories to attach to this file.
- **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 file with a label:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import Label
>>> c = CogniteClient()
>>> res = c.files.upload("/path/to/file", name="my_file", labels=[Label(external_
↳ id="WELL LOG")])
```

Upload a file with a geo_location:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import GeoLocation, Geometry
>>> c = CogniteClient()
```

(continues on next page)

(continued from previous page)

```
>>> geometry = Geometry(type="LineString", coordinates=[[30, 10], [10, 30], [40, 40],
↳ 40]])
>>> res = c.files.upload("/path/to/file", geo_location=GeoLocation(type="Feature",
↳ geometry=geometry))
```

Upload a string or bytes

`FilesAPI.upload_bytes` (*content: Union[str, bytes, TextIO, BinaryIO], name: str, external_id: Optional[str] = None, source: Optional[str] = None, mime_type: Optional[str] = None, metadata: Optional[Dict[str, str]] = None, directory: Optional[str] = None, asset_ids: Optional[Sequence[int]] = None, data_set_id: Optional[int] = None, labels: Optional[Sequence[cognite.client.data_classes.labels.Label]] = None, geo_location: Optional[cognite.client.data_classes.shared.GeoLocation] = None, source_created_time: Optional[int] = None, source_modified_time: Optional[int] = None, security_categories: Optional[Sequence[int]] = None, overwrite: bool = False*) → `cognite.client.data_classes.files.FileMetadata`

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.
- **directory** (*str*) – The directory to be associated with this file. Must be an absolute, unix-style path.
- **asset_ids** (*Sequence[int]*) – No description.
- **data_set_id** (*int*) – Id of the data set.
- **labels** (*Sequence[Label]*) – A list of the labels associated with this resource item.
- **geo_location** (*GeoLocation*) – The geographic metadata of the file.
- **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** (*Sequence[int]*) – Security categories to attach to this file.
- **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])
```

Retrieve download urls

`FilesAPI.retrieve_download_urls` (*id: Union[int, Sequence[int], None] = None, external_id: Union[str, Sequence[str], None] = None, extended_expiration: bool = False*) → Dict[Union[int, str], str]

Get download links by id or external id

Parameters

- **id** (*Union[int, Sequence[int]]*) – Id or list of ids.
- **external_id** (*Union[str, Sequence[str]]*) – External id or list of external ids.
- **extended_expiration** (*bool*) – Extend expiration time of download url to 1 hour. Defaults to false.

Returns Dictionary containing download urls.

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

Download files to disk

`FilesAPI.download` (*directory: Union[str, pathlib.Path], id: Union[int, Sequence[int], None] = None, external_id: Union[str, Sequence[str], None] = None*) → None

Download files by id or external id.

This method will stream all files to disk, never keeping more than 2MB in memory per worker. The files will be stored in the provided directory using the name retrieved from the file metadata in CDF.

Parameters

- **directory** (*str*) – Directory to download the file(s) to.
- **id** (*Union[int, Sequence[int]], optional*) – Id or list of ids
- **external_id** (*Union[str, Sequence[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: Union[pathlib.Path, str], id: Optional[int] = None, external_id: Optional[str] = None*) → 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: Optional[int] = None, external_id: Optional[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, Sequence[int], None] = None, external_id: Union[str, Sequence[str], None] = None*) → None

Delete files

Parameters

- `id` (`Union[int, Sequence[int]]`) – Id or list of ids
- `external_id` (`Union[str, Sequence[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, Sequence[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 Currently, a full replacement of labels on a file is not supported (only partial add/remove updates). See the example below on how to perform partial labels update.

Parameters *item* (`Union[FileMetadata, FileMetadataUpdate, Sequence[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)
```

Attach labels to a files:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import FileMetadataUpdate
>>> c = CogniteClient()
>>> my_update = FileMetadataUpdate(id=1).labels.add(["PUMP", "VERIFIED"])
>>> res = c.files.update(my_update)
```

Detach a single label from a file:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import FileMetadataUpdate
>>> c = CogniteClient()
>>> my_update = FileMetadataUpdate(id=1).labels.remove("PUMP")
>>> res = c.files.update(my_update)
```

Data classes

class `cognite.client.data_classes.files.FileAggregate` (*count*: *Optional[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*, *meta_data*: *Dict[str, str]* = *None*, *directory*: *str* = *None*, *asset_ids*: *Sequence[int]* = *None*, *data_set_id*: *int* = *None*, *labels*: *Sequence[Label]* = *None*, *geo_location*: *GeoLocation* = *None*, *source_created_time*: *int* = *None*, *source_modified_time*: *int* = *None*, *security_categories*: *Sequence[int]* = *None*, *id*: *int* = *None*, *uploaded*: *bool* = *None*, *uploaded_time*: *int* = *None*, *created_time*: *int* = *None*, *last_updated_time*: *int* = *None*, *cognite_client*: *CogniteClient* = *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.
- **directory** (*str*) – Directory associated with the file. Must be an absolute, unix-style path.
- **asset_ids** (*Sequence[int]*) – No description.
- **data_set_id** (*int*) – The dataSet Id for the item.
- **labels** (*Sequence[Label]*) – A list of the labels associated with this resource item.
- **geo_location** (*GeoLocation*) – The geographic metadata of the file.
- **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** (*Sequence[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: Se-
                                                         quence[int] = None,
                                                         asset_external_ids:
                                                         Sequence[str] =
                                                         None, data_set_ids:
                                                         Sequence[Dict[str,
                                                         Any]] = None, labels:
                                                         LabelFilter = None,
                                                         geo_location: GeoLo-
                                                         cationFilter = None,
                                                         asset_subtree_ids: Se-
                                                         quence[Dict[str, Any]]
                                                         = None, source: str =
                                                         None, created_time:
                                                         Union[Dict[str,
                                                         Any], Timestamp-
                                                         pRange] = None,
                                                         last_updated_time:
                                                         Union[Dict[str, Any],
                                                         TimestampRange] =
                                                         None, uploaded_time:
                                                         Union[Dict[str,
                                                         Any], Timestamp-
                                                         pRange] = None,
                                                         source_created_time:
                                                         Dict[str, Any] = None,
                                                         source_modified_time:
                                                         Dict[str, Any] = None,
                                                         external_id_prefix:
                                                         str = None, direc-
                                                         tory_prefix: str =
                                                         None, uploaded:
                                                         bool = None, cog-
                                                         nite_client: 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** (*Sequence[int]*) – Only include files that reference these specific asset IDs.
- **asset_external_ids** (*Sequence[str]*) – Only include files that reference these specific asset external IDs.

- **data_set_ids** (*Sequence[Dict[str, Any]]*) – Only include files that belong to these datasets.
- **labels** (*LabelFilter*) – Return only the files matching the specified label(s).
- **geo_location** (*GeoLocationFilter*) – Only include files matching the specified geographic relation.
- **asset_subtree_ids** (*Sequence[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.
- **directory_prefix** (*str*) – Filter by this (case-sensitive) prefix for the directory provided by the client.
- **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.

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]*

```
class cognite.client.data_classes.files.FileMetadataList (resources:      Collection[Any],
                                                         cognite_client: CogniteClient = None)
```

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

```
class cognite.client.data_classes.files.FileMetadataUpdate (id: Optional[int] = None,
                                                         external_id: Optional[str] = None)
```

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

Changes will be applied to file.

Args:

2.4.8 Functions

Create function

`FunctionsAPI.create` (*name*: *str*, *folder*: *Optional[str]* = *None*, *file_id*: *Optional[int]* = *None*, *function_path*: *str* = *'handler.py'*, *function_handle*: *Optional[Callable]* = *None*, *external_id*: *Optional[str]* = *None*, *description*: *Optional[str]* = *''*, *owner*: *Optional[str]* = *''*, *api_key*: *Optional[str]* = *None*, *secrets*: *Optional[Dict[KT, VT]]* = *None*, *env_vars*: *Optional[Dict[KT, VT]]* = *None*, *cpu*: *Optional[numbers.Number]* = *None*, *memory*: *Optional[numbers.Number]* = *None*, *runtime*: *Optional[str]* = *None*, *metadata*: *Optional[Dict[KT, VT]]* = *None*, *index_url*: *Optional[str]* = *None*, *extra_index_urls*: *Optional[List[str]]* = *None*) → `cognite.client.data_classes.functions.Function`

When creating a function, the source code can be specified in one of three ways:

- Via the *folder* argument, which is the path to the folder where the source code is located. *function_path* must point to a python file in the folder within which a function named *handle* must be defined.
- Via the *file_id* argument, which is the ID of a zip-file uploaded to the files API. *function_path* must point to a python file in the zipped folder within which a function named *handle* must be defined.
- Via the *function_handle* argument, which is a reference to a function object, which must be named *handle*.

The function named *handle* is the entrypoint of the created function. Valid arguments to *handle* are *data*, *client*, *secrets* and *function_call_info*:

- If the user calls the function with input data, this is passed through the *data* argument.
- If the user gives an *api_key* when creating the function, a pre instantiated `CogniteClient` is passed through the *client* argument.
- If the user gives one or more secrets when creating the function, these are passed through the *secrets* argument. The API key can be access through `secrets["apikey"]`.
- Data about the function call can be accessed via the argument *function_call_info*, which is a dictionary with keys *function_id* and, if the call is scheduled, *schedule_id* and *scheduled_time*.

Parameters

- **name** (*str*) – The name of the function.
- **folder** (*str*, *optional*) – Path to the folder where the function source code is located.
- **file_id** (*int*, *optional*) – File ID of the code uploaded to the Files API.
- **function_path** (*str*) – Relative path from the root folder to the file containing the *handle* function. Defaults to *handler.py*. Must be on POSIX path format.
- **function_handle** (*Callable*, *optional*) – Reference to a function object, which must be named *handle*.
- **external_id** (*str*, *optional*) – External id of the function.
- **description** (*str*, *optional*) – Description of the function.
- **owner** (*str*, *optional*) – Owner of this function. Typically used to know who created it.
- **api_key** (*str*, *optional*) – API key that can be used inside the function to access data in CDF.

- **secrets** (*Dict[str, str]*) – Additional secrets as key/value pairs. These can e.g. password to simulators or other data sources. Keys must be lowercase characters, numbers or dashes (-) and at most 15 characters. You can create at most 30 secrets, all keys must be unique, and cannot be apikey.
- **env_vars** (*Dict[str, str]*) – Environment variables as key/value pairs. Keys can contain only letters, numbers or the underscore character. You can create at most 100 environment variables.
- **cpu** (*Number, optional*) – Number of CPU cores per function. Allowed values are in the range [0.1, 0.6], and None translates to the API default which is 0.25 in GCP. The argument is unavailable in Azure.
- **memory** (*Number, optional*) – Memory per function measured in GB. Allowed values are in the range [0.1, 2.5], and None translates to the API default which is 1 GB in GCP. The argument is unavailable in Azure.
- **runtime** (*str, optional*) – The function runtime. Valid values are ["py37", "py38", "py39", None], and None translates to the API default which currently is "py38". The runtime "py38" resolves to the latest version of the Python 3.8 series.
- **metadata** (*Dict[str, str], optional*) – Metadata for the function as key/value pairs. Key & values can be at most 32, 512 characters long respectively. You can have at the most 16 key-value pairs, with a maximum size of 512 bytes.
- **index_url** (*str, optional*) – Index URL for Python Package Manager to use. Be aware of the intrinsic security implications of using the *index_url* option. [More information can be found on official docs](#),
- **extra_index_urls** (*List[str], optional*) – Extra Index URLs for Python Package Manager to use. Be aware of the intrinsic security implications of using the *extra_index_urls* option. [More information can be found on official docs](#),

Returns The created function.

Return type *Function*

Examples

Create function with source code in folder:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> function = c.functions.create(name="myfunction", folder="path/to/code",
↳function_path="path/to/function.py")
```

Create function with file_id from already uploaded source code:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> function = c.functions.create(name="myfunction", file_id=123, function_path=
↳"path/to/function.py")
```

Create function with predefined function object named *handle*:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> function = c.functions.create(name="myfunction", function_handle=handle)
```

Create function with predefined function object named *handle* with dependencies:

```
>>> from cognite.client import CogniteClient
>>>
>>> def handle(client, data):
>>>     """
>>>     [requirements]
>>>     numpy
>>>     [/requirements]
>>>     """
>>>     ...
>>>
>>> c = CogniteClient()
>>> function = c.functions.create(name="myfunction", function_handle=handle)
```

Note: When using a predefined function object, you can list dependencies between the tags *[requirements]* and *[/requirements]* in the function's docstring. The dependencies will be parsed and validated in accordance with requirement format specified in [PEP 508](#).

Delete function

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

Delete one or more functions.

Parameters

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

Returns None

Example

Delete functions by id or external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.functions.delete(id=[1,2,3], external_id="function3")
```

List functions

FunctionsAPI.**list**(*name: Optional[str] = None, owner: Optional[str] = None, file_id: Optional[int] = None, status: Optional[str] = None, external_id_prefix: Optional[str] = None, created_time: Union[Dict[str, int], cognite.client.data_classes.shared.TimestampRange, None] = None, limit: Optional[int] = 25*) → cognite.client.data_classes.functions.FunctionList

List all functions.

Parameters

- **name** (*str*) – The name of the function.

- **owner** (*str*) – Owner of the function.
- **file_id** (*int*) – The file ID of the zip-file used to create the function.
- **status** (*str*) – Status of the function. Possible values: [“Queued”, “Deploying”, “Ready”, “Failed”].
- **external_id_prefix** (*str*) – External ID prefix to filter on.
- **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.
- **limit** (*int*) – Maximum number of functions to return. Pass in -1, float(‘inf’) or None to list all.

Returns List of functions

Return type *FunctionList*

Example

List functions:

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

Retrieve function

FunctionsAPI.**retrieve** (*id: Optional[int] = None, external_id: Optional[str] = None*)
 → Union[cognite.client.data_classes.functions.FunctionList, cognite.client.data_classes.functions.Function, None]

Retrieve a single function by id.

Parameters

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

Returns Requested function or None if it does not exist.

Return type Optional[*Function*]

Examples

Get function by id:

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

Get function by external id:

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

Retrieve multiple functions

```
FunctionsAPI.retrieve_multiple(ids: Optional[Sequence[int]] = None, external_ids: Optional[Sequence[str]] = None) → Union[cognite.client.data_classes.functions.FunctionList, cognite.client.data_classes.functions.Function, None]
```

Retrieve multiple functions by id.

Parameters

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

Returns The requested functions.

Return type *FunctionList*

Examples

Get function by id:

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

Get functions by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.functions.retrieve_multiple(external_ids=["func1", "func2"])
```

Call function

```
FunctionsAPI.call(id: Optional[int] = None, external_id: Optional[str] = None, data: Optional[Dict[KT, VT]] = None, wait: bool = True) → cognite.client.data_classes.functions.FunctionCall
```

Call a function by its ID or external ID..

Parameters

- **id** (*int, optional*) – ID
- **external_id** (*str, optional*) – External ID
- **data** (*Union[str, dict], optional*) – Input data to the function (JSON serializable). This data is passed deserialized into the function through one of the arguments called data. **WARNING:** Secrets or other confidential information should not be passed via this argument. There is a dedicated *secrets* argument in `FunctionsAPI.create()` for this purpose.
- **wait** (*bool*) – Wait until the function call is finished. Defaults to True.

Returns A function call object.

Return type *FunctionCall*

Examples

Call a function by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> call = c.functions.call(id=1)
```

Call a function directly on the *Function* object:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> func = c.functions.retrieve(id=1)
>>> call = func.call()
```

Function calls

List function calls

`FunctionCallsAPI.list` (*function_id*: *Optional[int]* = *None*, *function_external_id*: *Optional[str]* = *None*, *status*: *Optional[str]* = *None*, *schedule_id*: *Optional[int]* = *None*, *start_time*: *Optional[Dict[str, int]]* = *None*, *end_time*: *Optional[Dict[str, int]]* = *None*, *limit*: *Optional[int]* = 25) → `cognite.client.data_classes.functions.FunctionCallList`

List all calls associated with a specific function id. Either *function_id* or *function_external_id* must be specified.

Parameters

- **function_id** (*int*, *optional*) – ID of the function on which the calls were made.
- **function_external_id** (*str*, *optional*) – External ID of the function on which the calls were made.
- **status** (*str*, *optional*) – Status of the call. Possible values [“Running”, “Failed”, “Completed”, “Timeout”].
- **schedule_id** (*int*, *optional*) – Schedule id from which the call belongs (if any).
- **start_time** (*Dict[str, int]*, *optional*) – Start time of the call. Possible keys are *min* and *max*, with values given as time stamps in ms.
- **end_time** (*Dict[str, int]*, *optional*) – End time of the call. Possible keys are *min* and *max*, with values given as time stamps in ms.
- **limit** (*int*, *optional*) – Maximum number of function calls to list. Pass in -1, float(‘inf’) or None to list all Function Calls.

Returns List of function calls

Return type *FunctionCallList*

Examples

List function calls:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> calls = c.functions.calls.list(function_id=1)
```

List function calls directly on a function object:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> func = c.functions.retrieve(id=1)
>>> calls = func.list_calls()
```

Retrieve function call

FunctionCallsAPI.**retrieve**(*call_id: int, function_id: Optional[int] = None, function_external_id: Optional[str] = None*) → Union[cognite.client.data_classes.functions.FunctionCallList, cognite.client.data_classes.functions.FunctionCall, None]

Retrieve a single function call by id.

Parameters

- **call_id**(*int*) – ID of the call.
- **function_id**(*int, optional*) – ID of the function on which the call was made.
- **function_external_id**(*str, optional*) – External ID of the function on which the call was made.

Returns Requested function call.

Return type Union[*FunctionCallList, FunctionCall, None*]

Examples

Retrieve single function call by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> call = c.functions.calls.retrieve(call_id=2, function_id=1)
```

Retrieve function call directly on a function object:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> func = c.functions.retrieve(id=1)
>>> call = func.retrieve_call(id=2)
```

Retrieve function call response

FunctionCallsAPI.**get_response**(*call_id: int, function_id: Optional[int] = None, function_external_id: Optional[str] = None*) → Dict[KT, VT]

Retrieve the response from a function call.

Parameters

- **call_id**(*int*) – ID of the call.
- **function_id**(*int, optional*) – ID of the function on which the call was made.
- **function_external_id**(*str, optional*) – External ID of the function on which the call was made.

Returns Response from the function call.

Examples

Retrieve function call response by call ID:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> response = c.functions.calls.get_response(call_id=2, function_id=1)
```

Retrieve function call response directly on a call object:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> call = c.functions.calls.retrieve(call_id=2, function_id=1)
>>> response = call.get_response()
```

Retrieve function call logs

`FunctionCallsAPI.get_logs` (*call_id: int, function_id: Optional[int] = None, function_external_id: Optional[str] = None*) → `cognite.client.data_classes.functions.FunctionCallLog`

Retrieve logs for function call.

Parameters

- **call_id** (*int*) – ID of the call.
- **function_id** (*int, optional*) – ID of the function on which the call was made.
- **function_external_id** (*str, optional*) – External ID of the function on which the call was made.

Returns Log for the function call.

Return type *FunctionCallLog*

Examples

Retrieve function call logs by call ID:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> logs = c.functions.calls.get_logs(call_id=2, function_id=1)
```

Retrieve function call logs directly on a call object:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> call = c.functions.calls.retrieve(call_id=2, function_id=1)
>>> logs = call.get_logs()
```

Function schedules

List function schedules

`FunctionSchedulesAPI.list` (*name*: `Optional[str] = None`, *function_id*: `Optional[int] = None`, *function_external_id*: `Optional[str] = None`, *created_time*: `Union[Dict[str, int], cognite.client.data_classes.shared.TimestampRange, None] = None`, *cron_expression*: `Optional[str] = None`, *limit*: `Optional[int] = 25`) → `cognite.client.data_classes.functions.FunctionSchedulesList`

List all schedules associated with a specific project.

Parameters

- **name** (*str*) – Name of the function schedule.
- **function_id** (*int*) – ID of the function the schedules are linked to.
- **function_external_id** (*str*) – External ID of the function the schedules are linked to.
- **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.
- **cron_expression** (*str*) – Cron expression.
- **limit** (*int*) – Maximum number of schedules to list. Pass in -1, float('inf') or None to list all.

Returns List of function schedules

Return type `FunctionSchedulesList`

Examples

List function schedules:

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

List schedules directly on a function object to get only schedules associated with this particular function:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> func = c.functions.retrieve(id=1)
>>> schedules = func.list_schedules(limit=None)
```

Create function schedule

`FunctionSchedulesAPI.create` (*name*: `str`, *cron_expression*: `str`, *function_id*: `Optional[int] = None`, *function_external_id*: `Optional[str] = None`, *client_credentials*: `Union[Dict[KT, VT], cognite.client.data_classes.iam.ClientCredentials, None] = None`, *description*: `str = ""`, *data*: `Optional[Dict[KT, VT]] = None`) → `cognite.client.data_classes.functions.FunctionSchedule`

Create a schedule associated with a specific project.

Parameters

- **name** (*str*) – Name of the schedule.
- **function_id** (*optional, int*) – Id of the function. This is required if the schedule is created with `client_credentials`.
- **function_external_id** (*optional, str*) – External id of the function. This is deprecated and cannot be used together with `client_credentials`.
- **description** (*str*) – Description of the schedule.
- **cron_expression** (*str*) – Cron expression.
- **client_credentials** – (optional, `ClientCredentials`, `Dict`): Instance of `ClientCredentials` or a dictionary containing client credentials: `client_id` `client_secret`
- **data** (*optional, Dict*) – Data to be passed to the scheduled run. **WARNING:** Secrets or other confidential information should not be passed via this argument. There is a dedicated `secrets` argument in `FunctionsAPI.create()` for this purpose.

Returns Created function schedule.

Return type `FunctionSchedule`

Examples

Create function schedule:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import ClientCredentials
>>> c = CogniteClient()
>>> schedule = c.functions.schedules.create(
...     name= "My schedule",
...     function_id=123,
...     cron_expression="*/5 * * * *",
...     client_credentials=ClientCredentials("my-client-id", "my-client-secret"),
...     description="This schedule does magic stuff."
... )
```

Delete function schedule

`FunctionSchedulesAPI.delete` (*id: int*) → `None`

Delete a schedule associated with a specific project.

Parameters `id` (*int*) – Id of the schedule

Returns `None`

Examples

Delete function schedule:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.functions.schedules.delete(id = 123)
```

Data classes

```
class cognite.client.data_classes.functions.Function (id: int = None, name: str = None, external_id: str = None, description: str = None, owner: str = None, status: str = None, file_id: int = None, function_path: str = None, created_time: int = None, api_key: str = None, secrets: Dict = None, env_vars: Dict = None, cpu: Number = None, memory: Number = None, runtime: str = None, runtime_version: str = None, metadata: Dict = None, error: Dict = None, cognite_client: CogniteClient = None)
```

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

A representation of a Cognite Function.

Parameters

- **id** (*int*) – Id of the function.
- **name** (*str*) – Name of the function.
- **external_id** (*str*) – External id of the function.
- **description** (*str*) – Description of the function.
- **owner** (*str*) – Owner of the function.
- **status** (*str*) – Status of the function.
- **file_id** (*int*) – File id of the code represented by this object.
- **function_path** (*str*) – Relative path from the root folder to the file containing the *handle* function. Defaults to *handler.py*. Must be on posix path format.
- **created_time** (*int*) – Created time in UNIX.
- **api_key** (*str*) – Api key attached to the function.
- **secrets** (*Dict[str, str]*) – Secrets attached to the function ((key, value) pairs).
- **env_vars** (*Dict[str, str]*) – User specified environment variables on the function ((key, value) pairs).
- **cpu** (*Number*) – Number of CPU cores per function. Defaults to 0.25. Allowed values are in the range [0.1, 0.6].
- **memory** (*Number*) – Memory per function measured in GB. Defaults to 1. Allowed values are in the range [0.1, 2.5].
- **runtime** (*str*) – Runtime of the function. Allowed values are [“py37”, “py38”, “py39”]. The runtime “py38” resolves to the latest version of the Python 3.8 series. Will default to “py38” if not specified.
- **runtime_version** (*str*) – The complete specification of the function runtime with major, minor and patch version numbers.

- **metadata** (*Dict[str, str]*) – Metadata associated with a function as a set of key:value pairs.
- **error** (*Dict[str, str]*) – Dictionary with keys “message” and “trace”, which is populated if deployment fails.
- **cognite_client** (*CogniteClient*) – An optional CogniteClient to associate with this data class.

call (*data: Optional[Dict[KT, VT]] = None, wait: bool = True*) → `cognite.client.data_classes.functions.FunctionCall`
 Call this particular function.

Parameters

- **data** (*Union[str, dict], optional*) – Input data to the function (JSON serializable). This data is passed deserialized into the function through one of the arguments called data. **WARNING:** Secrets or other confidential information should not be passed via this argument. There is a dedicated *secrets* argument in `FunctionsAPI.create()` for this purpose.
- **wait** (*bool*) – Wait until the function call is finished. Defaults to True.

Returns A function call object.

Return type *FunctionCall*

list_calls (*status: Optional[str] = None, schedule_id: Optional[int] = None, start_time: Optional[Dict[str, int]] = None, end_time: Optional[Dict[str, int]] = None, limit: Optional[int] = 25*) → `cognite.client.data_classes.functions.FunctionCallList`
 List all calls to this function.

Parameters

- **status** (*str, optional*) – Status of the call. Possible values [“Running”, “Failed”, “Completed”, “Timeout”].
- **schedule_id** (*int, optional*) – Schedule id from which the call belongs (if any).
- **start_time** (*[Dict[str, int], optional]*) – Start time of the call. Possible keys are *min* and *max*, with values given as time stamps in ms.
- **end_time** (*Dict[str, int], optional*) – End time of the call. Possible keys are *min* and *max*, with values given as time stamps in ms.
- **limit** (*int, optional*) – Maximum number of function calls to list. Pass in -1, float(“inf”) or None to list all Function Calls.

Returns List of function calls

Return type *FunctionCallList*

list_schedules (*limit: Optional[int] = 25*) → `cognite.client.data_classes.functions.FunctionSchedulesList`
 List all schedules associated with this function.

Parameters **limit** (*int*) – Maximum number of schedules to list. Pass in -1, float(“inf”) or None to list all.

Returns List of function schedules

Return type *FunctionSchedulesList*

retrieve_call (*id: int*) → `cognite.client.data_classes.functions.FunctionCall`
 Retrieve call by id.

Parameters **id** (*int*) – ID of the call.

Returns Function call.

Return type *FunctionCall*

update () → None

Update the function object. Can be useful to check for the latest status of the function ('Queued', 'Deploying', 'Ready' or 'Failed').

Returns None

```
class cognite.client.data_classes.functions.FunctionCall (id: int = None,
                                                    start_time: int = None,
                                                    end_time: int = None,
                                                    scheduled_time: int
                                                    = None, status: str =
                                                    None, schedule_id: int
                                                    = None, error: dict =
                                                    None, function_id: int
                                                    = None, cognite_client:
                                                    CogniteClient = None)
```

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

A representation of a Cognite Function call.

Parameters

- **id** (*int*) – A server-generated ID for the object.
- **start_time** (*int*) – Start time of the call, measured in number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **end_time** (*int*) – End time of the call, measured in number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **scheduled_time** (*int*) – Scheduled time of the call, measured in number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **status** (*str*) – Status of the function call (“Running”, “Completed” or “Failed”).
- **schedule_id** (*int*) – The schedule id belonging to the call.
- **error** (*dict*) – Error from the function call. It contains an error message and the stack trace.
- **cognite_client** (*CogniteClient*) – An optional CogniteClient to associate with this data class.

get_logs () → cognite.client.data_classes.functions.FunctionCallLog

Retrieve logs for this function call.

Returns Log for the function call.

Return type *FunctionCallLog*

get_response () → Dict[KT, VT]

Retrieve the response from this function call.

Returns Response from the function call.

update () → None

Update the function call object. Can be useful if the call was made with wait=False.

Returns None

```
class cognite.client.data_classes.functions.FunctionCallList (resources: Collection[Any],
                                                           cognite_client: CogniteClient =
                                                           None)
    Bases: cognite.client.data_classes._base.CogniteResourceList
```

```
class cognite.client.data_classes.functions.FunctionCallLog (resources: Collection[Any],
                                                             cognite_client: CogniteClient =
                                                             None)
    Bases: cognite.client.data_classes._base.CogniteResourceList
```

```
class cognite.client.data_classes.functions.FunctionCallLogEntry (timestamp: int = None,
                                                                    message: str = None, cognite_client: CogniteClient =
                                                                    None)
    Bases: cognite.client.data_classes._base.CogniteResource
```

A log entry for a function call.

Parameters

- **timestamp** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **message** (*str*) – Single line from stdout / stderr.

```
class cognite.client.data_classes.functions.FunctionList (resources: Collection[Any],
                                                           cognite_client: CogniteClient = None)
    Bases: cognite.client.data_classes._base.CogniteResourceList
```

```
class cognite.client.data_classes.functions.FunctionSchedule (id: int = None,
                                                                name: str = None,
                                                                function_id: str = None,
                                                                function_external_id: str = None,
                                                                description: str = None,
                                                                created_time: int = None,
                                                                cron_expression: str = None,
                                                                session_id: int = None,
                                                                when: str = None,
                                                                cognite_client: CogniteClient =
                                                                None)
    Bases: cognite.client.data_classes._base.CogniteResource
```

A representation of a Cognite Function Schedule.

Parameters

- **id** (*int*) – Id of the schedule.
- **name** (*str*) – Name of the function schedule.
- **function_id** (*int*) – Id of the function.
- **function_external_id** (*str*) – External id of the function.
- **description** (*str*) – Description of the function schedule.
- **cron_expression** (*str*) – Cron expression
- **created_time** (*int*) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.
- **session_id** (*int*) – ID of the session running with the schedule.
- **when** (*str*) – When the schedule will trigger, in human readable text (server generated from `cron_expression`).
- **cognite_client** (`CogniteClient`) – An optional `CogniteClient` to associate with this data class.

get_input_data () → Optional[dict]

Retrieve the input data to the associated function.

Returns Input data to the associated function or None if not set. This data is passed deserialized into the function through the data argument.

Return type Optional[Dict]

```
class cognite.client.data_classes.functions.FunctionSchedulesList (resources:  
                                                                    Collec-  
                                                                    tion[Any],  
                                                                    cog-  
                                                                    nite_client:  
                                                                    Cognite-  
                                                                    Client =  
                                                                    None)
```

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

```
class cognite.client.data_classes.functions.FunctionsLimits (timeout_minutes:  
                                                                    int,      cpu_cores:  
                                                                    Dict[str, float], mem-  
                                                                    ory_gb:      Dict[str,  
                                                                    float],      runtimes:  
                                                                    List[str],      re-  
                                                                    sponse_size_mb:  
                                                                    Optional[int] =  
                                                                    None)
```

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

Service limits for the associated project.

Parameters

- **timeout_minutes** (*int*) – Timeout of each function call.
- **cpu_cores** (`Dict[str, float]`) – The number of CPU cores per function execution (i.e. function call).

- **memory_gb** (*Dict[str, float]*) – The amount of available memory in GB per function execution (i.e. function call).
- **runtimes** (*List[str]*) – Available runtimes. For example, “py37” translates to the latest version of the Python 3.7.x series.
- **response_size_mb** (*Optional[int]*) – Maximum response size of function calls.

class `cognite.client.data_classes.functions.FunctionsStatus` (*status: str*)

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

Activation Status for the associated project.

Parameters **status** (*str*) – Activation Status for the associated project.

2.4.9 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[Sequence[int]] = None, external_ids: Optional[Sequence[str]] = None, ignore_unknown_ids: bool = False*) → `cognite.client.data_classes.time_series.TimeSeriesList`

Retrieve multiple time series by id.

Parameters

- **ids** (*Sequence[int], optional*) – IDs

- **external_ids** (*Sequence[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: Optional[str] = None, unit: Optional[str] = None, is_string: Optional[bool] = None, is_step: Optional[bool] = None, asset_ids: Optional[Sequence[int]] = None, asset_external_ids: Optional[Sequence[str]] = None, asset_subtree_ids: Union[int, Sequence[int], None] = None, asset_subtree_external_ids: Union[str, Sequence[str], None] = None, data_set_ids: Union[int, Sequence[int], None] = None, data_set_external_ids: Union[str, Sequence[str], None] = None, metadata: Optional[Dict[str, Any]] = None, external_id_prefix: Optional[str] = None, created_time: Optional[Dict[str, Any]] = None, last_updated_time: Optional[Dict[str, Any]] = None, partitions: Optional[int] = None, limit: int = 25*) → `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** (*Sequence[int], optional*) – List time series related to these assets.
- **asset_external_ids** (*Sequence[str], optional*) – List time series related to these assets.
- **asset_subtree_ids** (*Union[int, Sequence[int]]*) – Asset subtree id or list of asset subtree ids to filter on.
- **asset_subtree_external_ids** (*Union[str, Sequence[str]]*) – Asset external id or list of asset subtree external ids to filter on.

- **data_set_ids** (*Union[int, Sequence[int]]*) – Return only time series in the specified data set(s) with this id / these ids.
- **data_set_external_ids** (*Union[str, Sequence[str]]*) – Return only time series in the specified data set(s) with this external id / 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.

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] = 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*: Optional[str] = None, *description*: Optional[str] = None, *query*: Optional[str] = None, *filter*: Union[cognite.client.data_classes.time_series.TimeSeriesFilter, Dict[KT, VT], None] = 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, Sequence[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, Sequence[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, Sequence[int], None] = None`, *external_id*: `Union[str, Sequence[str], None] = None`, *ignore_unknown_ids*: `bool = False`) → `None`

Delete one or more time series.

Parameters

- **id** (`Union[int, Sequence[int]]`) – Id or list of ids
- **external_id** (`Union[str, Sequence[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, Sequence[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, Sequence[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: Sequence[int] = None, data_set_id: int = None, created_time: int = None, last_updated_time: int = None, legacy_name: str = None, cognite_client: CogniteClient = 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** (*Sequence[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`

Raises `ValueError` – If the time series is string as count aggregate is only supported for numeric data

Returns The total number of datapoints

Return type `int`

first () → `Optional[Datapoint]`

Returns the first datapoint in this time series. If empty, returns `None`.

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

Return type `Datapoint`

latest (*before: Union[int, str, datetime] = None*) → `Optional[Datapoint]`

Returns the latest datapoint in this time series. If empty, returns `None`.

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: Op-  
tional[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:
Sequence[int]
= None, as-
set_external_ids:
Sequence[str]
= None, as-
set_subtree_ids:
Se-
quence[Dict[str,
Any]] = None,
data_set_ids:
Se-
quence[Dict[str,
Any]] =
None, exter-
nal_id_prefix:
str = None,
created_time:
Union[Dict[str,
Any], Times-
tampRange]
= None,
last_updated_time:
Union[Dict[str,
Any], Times-
tampRange]
= None, cog-
nite_client:
CogniteClient =
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** (*Sequence[int]*) – Only include time series that reference these specific asset IDs.

- **asset_external_ids** (*Sequence[str]*) – Asset External IDs of related equipment that this time series relates to.
- **asset_subtree_ids** (*Sequence[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** (*Sequence[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: Collection[Any],  
cognite_client: CogniteClient = None)
```

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

```
class cognite.client.data_classes.time_series.TimeSeriesUpdate (id: Optional[int] = None, external_id: Optional[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.

2.4.10 Synthetic time series

Calculate the result of a function on time series

```
SyntheticDatapointsAPI.query (expressions: Union[str, sympy.Expr, Sequence[Union[str, sympy.Expr]]], start: Union[int, str, datetime], end: Union[int, str, datetime], limit: int = None, variables: Dict[str, Union[str, TimeSeries]] = None, aggregate: str = None, granularity: str = None) → Union[Datapoints, DatapointsList]
```

Calculate the result of a function on time series.

Parameters

- **expressions** (*Union[str, sympy.Expr, Sequence[Union[str, sympy.Expr]]]*) – Functions 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 per expression 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 DatapointsList object containing the calculated data.

Return type *Union[Datapoints, DatapointsList]*

Examples

Request a synthetic time series query with direct syntax

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> dps = c.time_series.data.synthetic.query(expressions="TS{id:123} + TS
↳{externalId:'abc'}", start="2w-ago", end="now")
```

Use variables to re-use an expression:

```
>>> vars = {"A": "my_ts_external_id", "B": client.time_series.retrieve(id=1)}
>>> dps = c.time_series.data.synthetic.query(expressions="A+B", start="2w-ago",
↳end="now", variables=vars)
```

Use sympy to build complex expressions:

```
>>> from sympy import symbols, cos, sin
>>> a = symbols('a')
>>> dps = c.time_series.data.synthetic.query([sin(a), cos(a)], start="2w-ago",
↳end="now", variables={"a": "my_ts_external_id"}, aggregate='interpolation',
↳granularity='1m')
```

2.4.11 Data points

Retrieve datapoints

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

Retrieve datapoints for one or more time series.

Performance guide: In order to retrieve millions of datapoints as efficiently as possible, here are a few guidelines:

1. For best speed, and significantly lower memory usage, consider using `retrieve_arrays(...)` which uses `numpy.ndarrays` for data storage.
2. Unlimited queries are fastest as they are trivial to parallelize. Thus, specifying a very large finite *limit*, e.g. 1 million, comes with a performance penalty.
3. Try to avoid specifying *start* and *end* to be very far from the actual data: If you have data from 2000 to 2015, don't set `start=0` (1970).

Parameters

- **start** (`Union[int, str, datetime, None]`) – Inclusive start. Default: 1970-01-01 UTC.
- **end** (`Union[int, str, datetime, None]`) – Exclusive end. Default: “now”
- **id** (`Union[None, int, Dict[str, Any], List[Union[int, Dict[str, Any]]]]`) – Id, dict (with id) or (mixed) sequence of these. See examples below.
- **external_id** (`Union[None, str, Dict[str, Any], List[Union[str, Dict[str, Any]]]]`) – External id, dict (with external id) or (mixed) sequence of these. See examples below.
- **aggregates** (`Union[str, List[str], None]`) – Single aggregate or list of aggregates to retrieve. Default: None (raw datapoints returned)
- **granularity** (`str`) – The granularity to fetch aggregates at. e.g. '15s', '2h', '10d'. Default: None.
- **limit** (`int`) – Maximum number of datapoints to return for each time series. Default: None (no limit)
- **include_outside_points** (`bool`) – Whether to include outside points. Not allowed when fetching aggregates. Default: False
- **ignore_unknown_ids** (`bool`) – Whether to ignore missing time series rather than raising an exception. Default: False

Returns A `Datapoints` object containing the requested data, or a `DatapointsList` if multiple time series were asked for (the ordering is `ids` first, then `external_ids`). If `ignore_unknown_ids` is `True`, a single time series is requested and it is not found, the function will return `None`.

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

Examples

You can specify the identifiers 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=42` from 2 weeks ago up until now:

```
>>> from cognite.client import CogniteClient
>>> client = CogniteClient()
>>> dps = client.time_series.data.retrieve(id=42, start="2w-ago")
```

You can also get aggregated values, such as *max* or *average*. You may also fetch more than one time series simultaneously. Here we are getting daily averages and maximum values for all of 2018, for two different time series, where we're specifying *start* and *end* as integers (milliseconds after epoch). Note that we are fetching them using their external ids:

```
>>> dps_lst = client.time_series.data.retrieve(
...     external_id=["foo", "bar"],
...     start=1514764800000,
...     end=1546300800000,
...     aggregates=["max", "average"],
...     granularity="1d")
```

In the two code examples above, we have a *dps* object (an instance of *Datapoints*), and a *dps_lst* object (an instance of *DatapointsList*). On *dps*, which in this case contains raw datapoints, you may access the underlying data directly by using the *.value* attribute. This works for both numeric and string (raw) datapoints, but not aggregates - they must be accessed by their respective names, because you're allowed to fetch up to 10 aggregates simultaneously, and they are stored on the same object:

```
>>> raw_data = dps.value
>>> first_dps = dps_lst[0] # optionally: `dps_lst.get(external_id="foo")`
>>> avg_data = first_dps.average
>>> max_data = first_dps.max
```

You may also slice a *Datapoints* object (you get *Datapoints* back), or ask for “a row of data” at a single index in same way you would do with a built-in *list* (you get a *Datapoint* object back, note the singular name). You'll also get *Datapoint* objects when iterating through a *Datapoints* object, but this should generally be avoided (consider this a performance warning):

```
>>> dps_slice = dps[-10:] # Last ten values
>>> dp = dps[3] # The third value
>>> for dp in dps_slice:
...     pass # do something!
```

All parameters can be individually set if you pass (one or more) dictionaries (even *ignore_unknown_ids*, contrary to the API). If you also pass top-level parameters, these will be overruled by the individual parameters (where both exist). You are free to mix any kind of ids and external ids: Single identifiers, single dictionaries and (mixed) lists of these.

Let's say you want different aggregates and end-times for a few time series (when only fetching a single aggregate, you may pass the string directly for convenience):

```
>>> dps_lst = client.time_series.data.retrieve(
...     id=[
...         {"id": 42, "end": "1d-ago", "aggregates": "average"},
...         {"id": 69, "end": "2d-ago", "aggregates": ["average"]},
...         {"id": 96, "end": "3d-ago", "aggregates": ["min", "max", "count"]},
...     ],
...     external_id={"external_id": "foo", "aggregates": "max"},
...     start="5d-ago",
...     granularity="1h")
```

When requesting multiple time series, an easy way to get the datapoints of a specific one is to use the *.get* method on the returned *DatapointsList* object, then specify if you want *id* or *external_id*. Note: If you fetch a time series by using *id*, you can still access it with its *external_id* (and the opposite way around), if you know it:

```
>>> from datetime import datetime, timezone
>>> utc = timezone.utc
>>> dps_lst = client.time_series.data.retrieve(
...     start=datetime(1907, 10, 14, tzinfo=utc),
...     end=datetime(1907, 11, 6, tzinfo=utc),
...     id=[42, 43, 44, ..., 499, 500],
```

(continues on next page)

(continued from previous page)

```
... )
>>> ts_350 = dps_lst.get(id=350) # `Datapoints` object
```

... but what happens if you request some duplicate ids or external_ids? In this example we will show how to get data from multiple disconnected periods. Let's say you're tasked to train a machine learning model to recognize a specific failure mode of a system, and you want the training data to only be from certain periods (when an alarm was on/high). Assuming these alarms are stored as events in CDF, with both start- and end times, we can use these directly in the query.

After fetching, the `.get` method will return a list of *Datapoints* instead, (assuming we have more than one event) in the same order, similar to how slicing works with non-unique indices on Pandas DataFrames:

```
>>> periods = client.events.list(type="alarm", subtype="pressure")
>>> sensor_xid = "foo-pressure-bar"
>>> dps_lst = client.time_series.data.retrieve(
...     id=[42, 43, 44],
...     external_id=[
...         {"external_id": sensor_xid, "start": ev.start_time, "end": ev.end_
↪time}
...         for ev in periods
...     ])
>>> ts_44 = dps_lst.get(id=44) # Single `Datapoints` object
>>> ts_lst = dps_lst.get(external_id=sensor_xid) # List of `len(periods)` ↪
↪`Datapoints` objects
```

The API has an endpoint to “retrieve latest (before)”, but not “after”. Luckily, we can emulate that behaviour easily. Let's say we have a very dense time series and do not want to fetch all of the available raw data (or fetch less precise aggregate data), just to get the very first datapoint of every month (from e.g. the year 2000 through 2010):

```
>>> import itertools
>>> month_starts = [
...     datetime(year, month, 1, tzinfo=utc)
...     for year, month in itertools.product(range(2000, 2011), range(1, 13))]
>>> dps_lst = client.time_series.data.retrieve(
...     external_id=[{"external_id": "foo", "start": start} for start in month_
↪starts],
...     limit=1)
```

To get *all* historic and future datapoints for a time series, e.g. to do a backup, you may want to import the two integer constants: `MIN_TIMESTAMP_MS` and `MAX_TIMESTAMP_MS`, to make sure you do not miss any. Performance warning: This pattern of fetching datapoints from the entire valid time domain is slower and shouldn't be used for regular “day-to-day” queries:

```
>>> from cognite.client.utils import MIN_TIMESTAMP_MS, MAX_TIMESTAMP_MS
>>> dps_backup = client.time_series.data.retrieve(
...     id=123,
...     start=MIN_TIMESTAMP_MS,
...     end=MAX_TIMESTAMP_MS + 1) # end is exclusive
```

The last example here is just to showcase the great flexibility of the `retrieve` endpoint, with a very custom query:

```
>>> ts1 = 1337
>>> ts2 = {
...     "id": 42,
...     "start": -12345, # Overrides `start` arg. below
```

(continues on next page)

(continued from previous page)

```

...     "end": "1h-ago",
...     "limit": 1000, # Overrides `limit` arg. below
...     "include_outside_points": True,
... }
>>> ts3 = {
...     "id": 11,
...     "end": "1h-ago",
...     "aggregates": ["max"],
...     "granularity": "42h",
...     "include_outside_points": False,
...     "ignore_unknown_ids": True, # Overrides `ignore_unknown_ids` arg. below
... }
>>> dps_lst = client.time_series.data.retrieve(
...     id=[ts1, ts2, ts3], start="2w-ago", limit=None, ignore_unknown_ids=False)

```

Retrieve datapoints as numpy arrays

`DatapointsAPI.retrieve_arrays` (*, *id*: Union[None, int, Dict[str, Any], Sequence[Union[int, Dict[str, Any]]]] = None, *external_id*: Union[None, str, Dict[str, Any], Sequence[Union[str, Dict[str, Any]]]] = None, *start*: Union[int, str, datetime.datetime, None] = None, *end*: Union[int, str, datetime.datetime, None] = None, *aggregates*: Union[str, List[str], None] = None, *granularity*: Optional[str] = None, *limit*: Optional[int] = None, *include_outside_points*: bool = False, *ignore_unknown_ids*: bool = False) → Union[None, cognite.client.data_classes.datapoints.DatapointsArray, cognite.client.data_classes.datapoints.DatapointsArrayList]

Retrieve datapoints for one or more time series.

Note: This method requires *numpy* to be installed.

Parameters

- **start** (Union[int, str, datetime, None]) – Inclusive start. Default: 1970-01-01 UTC.
- **end** (Union[int, str, datetime, None]) – Exclusive end. Default: “now”
- **id** (Union[None, int, Dict[str, Any], List[Union[int, Dict[str, Any]]]]) – Id, dict (with id) or (mixed) sequence of these. See examples below.
- **external_id** (Union[None, str, Dict[str, Any], List[Union[str, Dict[str, Any]]]]) – External id, dict (with external id) or (mixed) sequence of these. See examples below.
- **aggregates** (Union[str, List[str], None]) – Single aggregate or list of aggregates to retrieve. Default: None (raw datapoints returned)
- **granularity** (str) – The granularity to fetch aggregates at. e.g. '15s', '2h', '10d'. Default: None.
- **limit** (int) – Maximum number of datapoints to return for each time series. Default: None (no limit)
- **include_outside_points** (bool) – Whether to include outside points. Not allowed when fetching aggregates. Default: False

- **ignore_unknown_ids** (*bool*) – Whether to ignore missing time series rather than raising an exception. Default: `False`

Returns A *DatapointsArray* object containing the requested data, or a *DatapointsArrayList* if multiple time series were asked for (the ordering is *ids* first, then *external_ids*). If *ignore_unknown_ids* is *True*, a single time series is requested and it is not found, the function will return *None*.

Return type Union[None, *DatapointsArray*, *DatapointsArrayList*]

Examples

Note: For more usage examples, see *DatapointsAPI.retrieve* method (which accepts exactly the same arguments).

Get weekly *min* and *max* aggregates for a time series with *id=42* since the year 2000, then compute the range of values:

```
>>> from cognite.client import CogniteClient
>>> from datetime import datetime, timezone
>>> client = CogniteClient()
>>> dps = client.time_series.data.retrieve_arrays(
...     id=42,
...     start=datetime(2020, 1, 1, tzinfo=timezone.utc),
...     aggregates=["min", "max"],
...     granularity="7d")
>>> weekly_range = dps.max - dps.min
```

Get up-to 2 million raw datapoints for the last 48 hours for a noisy time series with *external_id="ts-noisy"*, then use a small and wide moving average filter to smooth it out:

```
>>> import numpy as np
>>> dps = client.time_series.data.retrieve_arrays(
...     external_id="ts-noisy",
...     start="2d-ago",
...     limit=2_000_000)
>>> smooth = np.convolve(dps.value, np.ones(5) / 5) # doctest: +SKIP
>>> smoother = np.convolve(dps.value, np.ones(20) / 20) # doctest: +SKIP
```

Get raw datapoints for multiple time series, that may or may not exist, from the last 2 hours, then find the largest gap between two consecutive values for all time series, also taking the previous value into account (outside point).

```
>>> id_lst = [42, 43, 44]
>>> dps_lst = client.time_series.data.retrieve_arrays(
...     id=id_lst,
...     start="2h-ago",
...     include_outside_points=True,
...     ignore_unknown_ids=True)
>>> largest_gaps = [np.max(np.diff(dps.timestamp)) for dps in dps_lst]
```

Get raw datapoints for a time series with *external_id="bar"* from the last 10 weeks, then convert to a *pandas.Series* (you can of course also use the *to_pandas()* convenience method if you want a *pandas.DataFrame*):

```
>>> import pandas as pd
>>> dps = client.time_series.data.retrieve_arrays(external_id="bar", start="10w-
↳ ago")
>>> series = pd.Series(dps.value, index=dps.timestamp)
```

Retrieve datapoints in pandas dataframe

```
DatapointsAPI.retrieve_dataframe(*, id: Union[None, int, Dict[str, Any], Sequence[Union[int, Dict[str, Any]]]] = None, external_id: Union[None, str, Dict[str, Any], Sequence[Union[str, Dict[str, Any]]]] = None, start: Union[int, str, datetime, None] = None, end: Union[int, str, datetime, None] = None, aggregates: Union[str, List[str], None] = None, granularity: Optional[str] = None, limit: Optional[int] = None, include_outside_points: bool = False, ignore_unknown_ids: bool = False, uniform_index: bool = False, include_aggregate_name: bool = True, include_granularity_name: bool = False, column_names: Literal['id', 'external_id'] = 'external_id') → pd.DataFrame
```

Get datapoints directly in a pandas dataframe.

Note: If you have duplicated time series in your query, the dataframe columns will also contain duplicates.

Parameters

- **start** (*Union[int, str, datetime, None]*) – Inclusive start. Default: 1970-01-01 UTC.
- **end** (*Union[int, str, datetime, None]*) – Exclusive end. Default: “now”
- **id** (*Union[None, int, Dict[str, Any], List[Union[int, Dict[str, Any]]]]*) – Id, dict (with id) or (mixed) sequence of these. See examples below.
- **external_id** (*Union[None, str, Dict[str, Any], List[Union[str, Dict[str, Any]]]]*) – External id, dict (with external id) or (mixed) sequence of these. See examples below.
- **aggregates** (*Union[str, List[str], None]*) – Single aggregate or list of aggregates to retrieve. Default: None (raw datapoints returned)
- **granularity** (*str*) – The granularity to fetch aggregates at. e.g. '15s', '2h', '10d'. Default: None.
- **limit** (*int*) – Maximum number of datapoints to return for each time series. Default: None (no limit)
- **include_outside_points** (*bool*) – Whether to include outside points. Not allowed when fetching aggregates. Default: False
- **ignore_unknown_ids** (*bool*) – Whether to ignore missing time series rather than raising an exception. Default: False
- **uniform_index** (*bool*) – If only querying aggregates AND a single granularity is used AND no limit is used, specifying *uniform_index=True* will return a dataframe with an equidistant datetime index from the earliest *start* to the latest *end* (missing values will be NaNs). If these requirements are not met, a ValueError is raised. Default: False
- **include_aggregate_name** (*bool*) – Include ‘aggregate’ in the column name, e.g. *my-ts|average*. Ignored for raw time series. Default: True
- **include_granularity_name** (*bool*) – Include ‘granularity’ in the column name, e.g. *my-ts|12h*. Added after ‘aggregate’ when present. Ignored for raw time series. Default: False
- **column_names** (*"id" | "external_id"*) – Use either ids or external ids as column names. Time series missing external id will use id as backup. Default: “external_id”

Returns A pandas DataFrame containing the requested time series. The ordering of columns is ids first, then external_ids. For time series with multiple aggregates, they will be sorted in alphabetical order (“average” before “max”).

Return type pandas.DataFrame

Examples

Get a pandas dataframe using a single id, and use this id as column name, with no more than 100 datapoints:

```
>>> from cognite.client import CogniteClient
>>> client = CogniteClient()
>>> df = client.time_series.data.retrieve_dataframe(
...     id=12345,
...     start="2w-ago",
...     end="now",
...     limit=100,
...     column_names="id")
```

Get the pandas dataframe with a uniform index (fixed spacing between points) of 1 day, for two time series with individually specified aggregates, from 1990 through 2020:

```
>>> from datetime import datetime, timezone
>>> df = client.time_series.data.retrieve_dataframe(
...     id=[
...         {"external_id": "foo", "aggregates": ["discrete_variance"]},
...         {"external_id": "bar", "aggregates": ["total_variation", "continuous_
↪variance"]},
...     ],
...     granularity="1d",
...     start=datetime(1990, 1, 1, tzinfo=timezone.utc),
...     end=datetime(2020, 12, 31, tzinfo=timezone.utc),
...     uniform_index=True)
```

Get a pandas dataframe containing the ‘average’ aggregate for two time series using a 30-day granularity, starting Jan 1, 1970 all the way up to present, without having the aggregate name in the column names:

```
>>> df = client.time_series.data.retrieve_dataframe(
...     external_id=["foo", "bar"],
...     aggregates=["average"],
...     granularity="30d",
...     include_aggregate_name=False)
```

Retrieve datapoints in time zone in pandas dataframe

DatapointsAPI.`retrieve_dataframe_in_tz`(*`id`: `int` | `Sequence[int]` | `None` = `None`, `external_id`: `str` | `Sequence[str]` | `None` = `None`, `start`: `datetime`, `end`: `datetime`, `aggregates`: `Sequence[str]` | `str` | `None` = `None`, `granularity`: `Optional[str]` = `None`, `ignore_unknown_ids`: `bool` = `False`, `uniform_index`: `bool` = `False`, `include_aggregate_name`: `bool` = `True`, `include_granularity_name`: `bool` = `False`, `column_names`: `Literal['id', 'external_id']` = `'external_id'`) → `pd.DataFrame`

Get datapoints directly in a pandas dataframe in the same time zone as start and end.

Note: This is a convenience method. It builds on top of the methods `retrieve_arrays` and `retrieve_dataframe`. It enables you to get correct aggregates in your local time zone with daily, weekly, monthly, quarterly, and yearly aggregates with automatic handling for daylight saving time (DST) transitions. If your time zone observes DST, and your query crosses at least one DST-boundary, granularities like “3 days” or “1 week”, that used to represent fixed durations, no longer do so. To understand why, let’s illustrate with an example: A typical time zone that observes DST will skip one hour ahead during spring, leading to a day that is only 23 hours long, and oppositely in the fall, turning back the clock one hour, yielding a 25-hour long day.

In short, this method works as follows:

1. Get the time zone from start and end (must be equal).
2. Split the time range from start to end into intervals based on DST boundaries.
3. Create a query for each interval and pass all to the `retrieve_arrays` method.
4. Stack the resulting arrays into a single column in the resulting DataFrame.

Warning: The queries to `retrieve_arrays` are translated to a multiple of hours. This means that time zones that are not a whole hour offset from UTC are not supported (yet). The same is true for time zones that observe DST with an offset from standard time that is not a multiple of 1 hour.

Parameters

- **id** (*int* | *Sequence[int]* | *None*) – ID or list of IDs.
- **external_id** (*str* | *Sequence[str]* | *None*) – External ID or list of External IDs.
- **start** (*datetime*) – Inclusive start, must be time zone aware.
- **end** (*datetime*) – Exclusive end, must be time zone aware and have the same time zone as start.
- **aggregates** (*str* | *list[str]* | *None*) – Single aggregate or list of aggregates to retrieve. Default: None (raw datapoints returned)
- **granularity** (*str*) – The granularity to fetch aggregates at, supported are: second, minute, hour, day, week, month, quarter and year. Default: None.
- **ignore_unknown_ids** (*bool*) – Whether to ignore missing time series rather than raising an exception. Default: False
- **uniform_index** (*bool*) – If querying aggregates, specifying *uniform_index=True* will return a dataframe with an index with constant spacing between timestamps decided by granularity all the way from *start* to *end* (missing values will be NaNs). Default: False
- **include_aggregate_name** (*bool*) – Include ‘aggregate’ in the column name, e.g. *my-tslaverage*. Ignored for raw time series. Default: True
- **include_granularity_name** (*bool*) – Include ‘granularity’ in the column name, e.g. *my-tsl12h*. Added after ‘aggregate’ when present. Ignored for raw time series. Default: False
- **column_names** ("*id*" | "*external_id*") – Use either ids or external ids as column names. Time series missing external id will use id as backup. Default: “external_id”

Returns A pandas DataFrame containing the requested time series with a DatetimeIndex localized in the given time zone.

Return type pandas.DataFrame

Examples

Get a pandas dataframe in the time zone of Oslo, Norway:

```
>>> from cognite.client import CogniteClient
>>> # In Python >=3.9 you may import directly from `zoneinfo`
>>> from cognite.client.utils import ZoneInfo
>>> client = CogniteClient()
>>> df = client.time_series.data.retrieve_dataframe_in_tz(
...     id=12345,
...     start=datetime(2023, 1, 1, tzinfo=ZoneInfo("Europe/Oslo")),
...     end=datetime(2023, 2, 1, tzinfo=ZoneInfo("Europe/Oslo")),
...     aggregates="average",
...     granularity="1week",
...     column_names="id")
```

Get a pandas dataframe with the sum and continuous variance of the time series with external id “foo” and “bar”, for each quarter from 2020 to 2022 returned in the time zone of Oslo, Norway:

```
>>> from cognite.client import CogniteClient
>>> # In Python >=3.9 you may import directly from `zoneinfo`
>>> from cognite.client.utils import ZoneInfo
>>> client = CogniteClient()
>>> df = client.time_series.data.retrieve_dataframe(
...     external_id=["foo", "bar"],
...     aggregates=["sum", "continuous_variance"],
...     granularity="1quarter",
...     start=datetime(2020, 1, 1, tzinfo=ZoneInfo("Europe/Oslo")),
...     end=datetime(2022, 12, 31, tzinfo=ZoneInfo("Europe/Oslo")))
```

Tip: You can also use shorter granularities such as second(s), minute(s), hour(s), which do not require any special handling of DST. The longer granularities at your disposal, which are adjusted for DST, are: day(s), week(s), month(s), quarter(s) and year(s). All the granularities support a one-letter version s, m, h, d, w, q, and y, except for month, to avoid confusion with minutes. Furthermore, the granularity is expected to be given as a lowercase.

Retrieve latest datapoint

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

Get the latest datapoint for one or more time series

Parameters

- **id** (`Union[int, LatestDatapointQuery, List[Union[int, LatestDatapointQuery]]]`) – Id or list of ids.
- **external_id** (`Union[str, LatestDatapointQuery, List[Union[str, LatestDatapointQuery]]]`) – External id or list of external ids.
- **before** – (`Union[int, str, datetime]`): Get latest datapoint before this time. Not used when passing ‘LatestDatapointQuery’.
- **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 DatapointsList if multiple were requested. If `ignore_unknown_ids` is `True`, a single time series is requested and it is not found, the function will return `None`.

Return type `Union[None, 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.time_series.data.retrieve_latest(id=1)[0]
```

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

```
>>> res = c.time_series.data.retrieve_latest(id=1, before="2d-ago")[0]
```

You may also pass an instance of LatestDatapointQuery:

```
>>> from cognite.client.data_classes import LatestDatapointQuery
>>> res = c.time_series.data.retrieve_latest(id=LatestDatapointQuery(id=1,
↳before=60_000))[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:

```
>>> res = c.time_series.data.retrieve_latest(external_id=["abc", "def"])
>>> latest_abc = res[0][0]
>>> latest_def = res[1][0]
```

If you need to specify a different value of ‘before’ for each time series, you may pass several LatestDatapoint-Query objects:

```
>>> from datetime import datetime, timezone
>>> id_queries = [
...     123,
...     LatestDatapointQuery(id=456, before="1w-ago"),
...     LatestDatapointQuery(id=789, before=datetime(2018,1,1, tzinfo=timezone.
↳utc))]
>>> res = c.time_series.data.retrieve_latest(
...     id=id_queries,
...     external_id=LatestDatapointQuery(external_id="abc", before="3h-ago"))
```

Insert data points

`DatapointsAPI.insert` (*datapoints*: `Union[cognite.client.data_classes.datapoints.Datapoints, cognite.client.data_classes.datapoints.DatapointsArray, Sequence[Dict[str, Union[int, float, str, datetime.datetime]]], Sequence[Tuple[Union[int, float, datetime.datetime], Union[int, float, str]]]]`, *id*: `Optional[int] = None`, *external_id*: `Optional[str] = None`) → `None`

Insert datapoints into a time series

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

Parameters

- **datapoints** (`Union[Datapoints, DatapointsArray, List[Dict], List[Tuple]]`) – The datapoints you wish to insert. Can either be a list of tuples, a list of dictionaries, a `Datapoints` object or a `DatapointsArray` 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
>>> from datetime import datetime, timezone
>>> c = CogniteClient()
>>> # With datetime objects:
>>> datapoints = [
...     (datetime(2018,1,1, tzinfo=timezone.utc), 1000),
...     (datetime(2018,1,2, tzinfo=timezone.utc), 2000),
... ]
>>> c.time_series.data.insert(datapoints, id=1)
>>> # With ms since epoch:
>>> datapoints = [(1500000000000, 1000), (1600000000000, 2000)]
>>> c.time_series.data.insert(datapoints, id=2)
```

Or they can be a list of dictionaries:

```
>>> datapoints = [
...     {"timestamp": 1500000000000, "value": 1000},
...     {"timestamp": 1600000000000, "value": 2000},
... ]
>>> c.time_series.data.insert(datapoints, external_id="abcd")
```

Or they can be a `Datapoints` or `DatapointsArray` object (with raw datapoints only). Note that the `id` or `external_id` set on these objects are not inspected/used (as they belong to the “from-time-series”, and not the “to-time-series”), and so you must explicitly pass the identifier of the time series you want to insert into, which in this example is `external_id="foo"`:

```
>>> data = c.time_series.data.retrieve(external_id="abc", start="1w-ago", end="now")
>>> c.time_series.data.insert(data, external_id="foo")
```

Insert data points into multiple time series

`DatapointsAPI.insert_multiple` (*datapoints*: `List[Dict[str, Union[str, int, List[T], cognite.client.data_classes.datapoints.Datapoints, cognite.client.data_classes.datapoints.DatapointsArray]]]`) → `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 dictionaries, each containing datapoints for a different (presumably) time series. These dictionaries must have the key “datapoints” (containing the data) specified as a `Datapoints` object, a `DatapointsArray` object, or list of either tuples (`timestamp, value`) or dictionaries, {“timestamp”: `ts`, “value”: `value`}:

```
>>> from cognite.client import CogniteClient
>>> from datetime import datetime, timezone
>>> client = CogniteClient()

>>> datapoints = []
>>> # With datetime objects and id
>>> datapoints.append(
...     {"id": 1, "datapoints": [
...         (datetime(2018,1,1,tzinfo=timezone.utc), 1000),
...         (datetime(2018,1,2,tzinfo=timezone.utc), 2000),
...     ]})

>>> # With ms since epoch and external_id:
>>> datapoints.append({"external_id": "foo", "datapoints": [(1500000000000, 1000),
↪ (1600000000000, 2000)]})

>>> # With raw data in a Datapoints object (or DatapointsArray):
>>> data_to_clone = client.time_series.data.retrieve(external_id="bar")
>>> datapoints.append({"external_id": "bar-clone", "datapoints": data_to_clone})
>>> client.time_series.data.insert_multiple(datapoints)
```

Insert pandas dataframe

`DatapointsAPI.insert_dataframe` (*df*: `pd.DataFrame`, *external_id_headers*: `bool = True`, *dropna*: `bool = True`) → `None`

Insert a dataframe (columns must be unique).

The index of the dataframe must contain the timestamps (`pd.DatetimeIndex`). The names of the columns specify the ids or external ids of the time series to which the datapoints will be written.

Said time series must already exist.

Parameters

- **df** (`pandas.DataFrame`) – Pandas `DataFrame` object containing the time series.
- **external_id_headers** (`bool`) – Interpret the column names as external id. Pass `False` if using ids. Default: `True`.

- **dropna** (*bool*) – Set to True to ignore NaNs in the given DataFrame, applied per column. Default: True.

Returns None

Examples

Post a dataframe with white noise:

```
>>> import numpy as np
>>> import pandas as pd
>>> from cognite.client import CogniteClient
>>>
>>> client = CogniteClient()
>>> ts_xid = "my-foo-ts"
>>> idx = pd.date_range(start="2018-01-01", periods=100, freq="1d")
>>> noise = np.random.normal(0, 1, 100)
>>> df = pd.DataFrame({ts_xid: noise}, index=idx)
>>> client.time_series.data.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: Optional[int] = None, external_id: Optional[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]*) – Exclusive 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.time_series.data.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 list of datapoint ids along with time range to delete. See examples below.

Returns None

Examples

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

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

Data classes

```
class cognite.client.data_classes.datapoints.Datapoint (timestamp: Optional[int]
= None, value: Union[str, float, None] = None, average: Optional[float]
= None, max: Optional[float] = None, min: Optional[float] = None, count: Optional[int]
= None, sum: Optional[float] = None, interpolation: Optional[float] = None,
step_interpolation: Optional[float] = None, continuous_variance:
Optional[float] = None, discrete_variance: Optional[float] = None,
total_variation: Optional[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). Can be negative to define a date before 1970. Minimum timestamp is 1900.01.01 00:00:00 UTC
- **value** (`Union[str, float]`) – The data value. Can be string or numeric
- **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: bool = False*) → `pandas.DataFrame`
Convert the datapoint into a pandas DataFrame.

Parameters **camel_case** (*bool*) – Convert column names to camel case (e.g. *stepInterpolation* instead of *step_interpolation*)

Returns `pandas.DataFrame`

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

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

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

- **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.
- **granularity** (*str*) – The granularity of the aggregate datapoints (does not apply to raw data)
- **timestamp** (*List[int]*) – The data timestamps in milliseconds since the epoch (Jan 1, 1970). Can be negative to define a date before 1970. Minimum timestamp is 1900.01.01 00:00:00 UTC
- **value** (*Union[List[str], List[float]]*) – The data values. Can be string or numeric
- **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 dictionary representing the instance.

Return type Dict[str, Any]

to_pandas (*column_names: str = 'external_id', include_aggregate_name: bool = True, include_granularity_name: bool = False, 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 “external_id”, can also be “id”. For time series with no external ID, ID will be used instead.
- **include_aggregate_name** (*bool*) – Include aggregate in the column name
- **include_granularity_name** (*bool*) – Include granularity in the column name (after aggregate if present)
- **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.DatapointsArray (id: int = None,  
external_id: str =  
None, is_string:  
bool = None,  
is_step: bool =  
None, unit: str =  
None, granularity:  
str = None, times-  
tamp: NumpyDate-  
time64NSArray  
= None, value:  
Union[NumpyFloat64Array,  
NumpyObjArray]  
= None, average:  
NumpyFloat64Array  
= None, max:  
NumpyFloat64Array  
= None, min:  
NumpyFloat64Array  
= None, count:  
NumpyInt64Array  
= None, sum:  
NumpyFloat64Array  
= None, in-  
terpolation:  
NumpyFloat64Array  
= None,  
step_interpolation:  
NumpyFloat64Array  
= None, contin-  
uous_variance:  
NumpyFloat64Array  
= None, dis-  
crete_variance:  
NumpyFloat64Array  
= None, to-  
tal_variation:  
NumpyFloat64Array  
= None)
```

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

An object representing datapoints using numpy arrays.

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

Dump the DatapointsArray into a json serializable Python data type.

Parameters

- **camel_case** (*bool*) – Use camelCase for attribute names. Default: False.
- **convert_timestamps** (*bool*) – Convert timestamps to ISO 8601 formatted strings. Default: False (returns as integer, milliseconds since epoch)

Returns A dictionary representing the instance.

Return type Dict[str, Any]

to_pandas (*column_names: Literal[('id', 'external_id')] = 'external_id', include_aggregate_name: bool = True, include_granularity_name: bool = False*) → pandas.DataFrame
Convert the DatapointsArray into a pandas DataFrame.

Parameters

- **column_names** (*str*) – Which field to use as column header. Defaults to “external_id”, can also be “id”. For time series with no external ID, ID will be used instead.
- **include_aggregate_name** (*bool*) – Include aggregate in the column name
- **include_granularity_name** (*bool*) – Include granularity in the column name (after aggregate if present)

Returns The datapoints as a pandas DataFrame.

Return type pandas.DataFrame

```
class cognite.client.data_classes.datapoints.DatapointsArrayList (resources: Collection[Any], cognite_client: CogniteClient = None)
```

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

concat_duplicate_ids () → None

Concatenates all arrays with duplicated IDs.

Arrays with the same ids are stacked in chronological order.

Caveat This method is not guaranteed to preserve the order of the list.

dump (*camel_case: bool = False, convert_timestamps: 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. Default: False.
- **convert_timestamps** (*bool*) – Convert timestamps to ISO 8601 formatted strings. Default: False (returns as integer, milliseconds since epoch)

Returns A list of dicts representing the instance.

Return type List[Dict[str, Any]]

get (*id: Optional[int] = None, external_id: Optional[str] = None*) → Union[None, cognite.client.data_classes.datapoints.DatapointsArray, List[cognite.client.data_classes.datapoints.DatapointsArray]]
Get a specific DatapointsArray from this list by id or external_id.

Note: For duplicated time series, returns a list of DatapointsArray.

Parameters

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

Returns The requested item(s)

Return type Union[None, *DatapointsArray*, List[*DatapointsArray*]]

to_pandas (*column_names*: Literal[('id', 'external_id')] = 'external_id', *include_aggregate_name*: bool = True, *include_granularity_name*: bool = False) → pandas.DataFrame
Convert the DatapointsArrayList into a pandas DataFrame.

Parameters

- **column_names** (*str*) – Which field to use as column header. Defaults to “external_id”, can also be “id”. For time series with no external ID, ID will be used instead.
- **include_aggregate_name** (*bool*) – Include aggregate in the column name
- **include_granularity_name** (*bool*) – Include granularity in the column name (after aggregate if present)

Returns The datapoints as a pandas DataFrame.

Return type pandas.DataFrame

```
class cognite.client.data_classes.datapoints.DatapointsList (resources: Collection[Any],
                                                            cognite_client: CogniteClient = None)
```

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

```
get (id: Optional[int] = None, external_id: Optional[str] = None)
    → Union[None, cognite.client.data_classes.datapoints.Datapoints,
            List[cognite.client.data_classes.datapoints.Datapoints]]
Get a specific Datapoints from this list by id or external_id.
```

Note: For duplicated time series, returns a list of Datapoints.

Parameters

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

Returns The requested item(s)

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

to_pandas (*column_names*: Literal[('id', 'external_id')] = 'external_id', *include_aggregate_name*: bool = True, *include_granularity_name*: bool = False) → pandas.DataFrame
Convert the datapoints list into a pandas DataFrame.

Parameters

- **column_names** (*str*) – Which field to use as column header. Defaults to “external_id”, can also be “id”. For time series with no external ID, ID will be used instead.
- **include_aggregate_name** (*bool*) – Include aggregate in the column name
- **include_granularity_name** (*bool*) – Include granularity in the column name (after aggregate if present)

Returns The datapoints list as a pandas DataFrame.

Return type pandas.DataFrame

```
class cognite.client.data_classes.datapoints.LatestDatapointQuery (id: Optional[int] = None, external_id: Optional[str] = None, before: Union[int, str, datetime.datetime, None] = None)
```

Bases: object

Parameters describing a query for the latest datapoint from a time series.

Note: Pass either ID or external ID.

Parameters

- **id** (*Optional[int]*) – The internal ID of the time series to query.
- **external_id** (*Optional[str]*) – The external ID of the time series to query.
- **before** (*Union[None, int, str, datetime]*) – Get latest datapoint before this time. None means ‘now’.

2.4.12 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[Sequence[int]] = None*, *external_ids*: *Optional[Sequence[str]] = None*, *ignore_unknown_ids*: *bool = False*) → `cognite.client.data_classes.sequences.SequenceList`

Retrieve multiple sequences by id.

Parameters

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

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*: *Optional[str] = None*, *external_id_prefix*: *Optional[str] = None*, *metadata*: *Optional[Dict[str, str]] = None*, *asset_ids*: *Optional[Sequence[int]] = None*, *asset_subtree_ids*: *Union[int, Sequence[int], None] = None*, *asset_subtree_external_ids*: *Union[str, Sequence[str], None] = None*, *data_set_ids*: *Union[int, Sequence[int], None] = None*, *data_set_external_ids*: *Union[str, Sequence[str], None] = None*, *created_time*: *Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None*, *last_updated_time*: *Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = 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** (*SequenceType[int]*) – Filter out sequences that are not linked to any of these assets.
- **asset_subtree_ids** (*Union[int, SequenceType[int]]*) – Asset subtree id or list of asset subtree ids to filter on.
- **asset_subtree_external_ids** (*Union[str, SequenceType[str]]*) – Asset subtree external id or list of asset subtree external ids to filter on.
- **data_set_ids** (*Union[int, SequenceType[int]]*) – Return only sequences in the specified data set(s) with this id / these ids.
- **data_set_external_ids** (*SequenceType[str]*) – Return only sequences in the specified data set(s) with this external id / 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] = 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*: Optional[str] = None, *description*: Optional[str] = None, *query*: Optional[str] = None, *filter*: Union[cognite.client.data_classes.sequences.SequenceFilter, Dict[KT, VT], None] = 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, Sequence[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, SequenceType[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, Sequence[int], None] = None, *external_id*: Union[str, Sequence[str], None] = None) → None

Delete one or more sequences.

Parameters

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

Updating column definitions

Currently, updating the column definitions of a sequence is only supported through partial update, using `add`, `remove` and `modify` methods on the `columns` property.

Add a single new column:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import SequenceUpdate
>>> c = CogniteClient()
>>>
>>> my_update = SequenceUpdate(id=1).columns.add({"valueType": "STRING", "externalId
↳ ": "user", "description": "some description"})
>>> res = c.sequences.update(my_update)
```

Add multiple new columns:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import SequenceUpdate
>>> c = CogniteClient()
>>>
>>> column_def = [{"valueType":"STRING","externalId":"user","description":"some_
↳description"}, {"valueType":"DOUBLE","externalId":"amount"}]
>>> my_update = SequenceUpdate(id=1).columns.add(column_def)
>>> res = c.sequences.update(my_update)
```

Remove a single column:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import SequenceUpdate
>>> c = CogniteClient()
>>>
>>> my_update = SequenceUpdate(id=1).columns.remove("col_external_id1")
>>> res = c.sequences.update(my_update)
```

Remove multiple columns:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import SequenceUpdate
>>> c = CogniteClient()
>>>
>>> my_update = SequenceUpdate(id=1).columns.remove(["col_external_id1","col_
↳external_id2"])
>>> res = c.sequences.update(my_update)
```

Update existing columns:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import SequenceUpdate, SequenceColumnUpdate
>>> c = CogniteClient()
>>>
>>> column_updates = [
...     SequenceColumnUpdate(external_id="col_external_id_1").external_id.set(
↳"new_col_external_id"),
...     SequenceColumnUpdate(external_id="col_external_id_2").description.set("my_
↳new description"),
... ]
>>> my_update = SequenceUpdate(id=1).columns.modify(column_updates)
>>> res = c.sequences.update(my_update)
```

Retrieve data

`SequencesDataAPI.retrieve` (*start*: `int`, *end*: `Optional[int]`, *column_external_ids*: `Optional[Sequence[str]] = None`, *external_id*: `Union[str, Sequence[str], None] = None`, *id*: `Union[int, Sequence[int], None] = None`, *limit*: `Optional[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[SequenceType[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. 10000 is the maximum limit per request.

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: Union[int, None], column_external_ids: Optional[List[str]] = None, external_id: str = None, column_names: str = None, id: int = None, limit: int = None*) → `pandas.DataFrame`

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[SequenceType[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, Sequence[Union[int, float, str]]], Sequence[Tuple[int, Sequence[Union[int, float, str]]], Sequence[Dict[str, Any]], cognite.client.data_classes.sequences.SequenceData], column_external_ids: Optional[Sequence[str]], id: Optional[int] = None, external_id: Optional[str] = None) → None`

Insert rows into a sequence

Parameters

- **column_external_ids** (`Optional[SequenceType[str]]`) – List of external id for the columns of the sequence.
- **rows** (`Union[Dict[int, SequenceType[Union[int, float, str]]], SequenceType[Tuple[int, SequenceType[Union[int, float, str]]], SequenceType[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 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: pandas.DataFrame, 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.
- **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: Sequence[int], id: Optional[int] = None, external_id: Optional[str] = None*) → None

Delete rows from a sequence

Parameters

- **rows** (*SequenceType[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: Optional[int] = None, external_id: Optional[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: SequenceType[Dict[str, Any]] = None, created_time: int = None, last_updated_time: int = None, data_set_id: int = None, cognite_client: CogniteClient = None)
```

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** (*SequenceType[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.

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

Bases: dict

No description.

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

```
class cognite.client.data_classes.sequences.SequenceColumnUpdate (id:      Optional[int]      = None,  
                                                                external_id:    Optional[str]    = None)
```

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

No description.

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

```
class cognite.client.data_classes.sequences.SequenceData (id:      Optional[int]      = None,  
                                                                external_id:    Optional[str]    = None,  
                                                                rows:          Optional[Sequence[dict]] = None,  
                                                                row_numbers:  Optional[Sequence[int]] = None,  
                                                                values:       Optional[Sequence[Sequence[Union[int, float, str]]]] = None,  
                                                                columns:     Optional[Sequence[Dict[str, Any]]] = None)
```

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

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** (*SequenceType[dict]*) – Combined row numbers and row data object from the API. If you pass this, row_numbers/values are ignored.
- **row_numbers** (*SequenceType[int]*) – The data row numbers.
- **values** (*SequenceType[SequenceType[Union[int, str, float]]]*) – The data values, one row at a time.
- **columns** – *SequenceType[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 dictionary representing the instance.

Return type *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: Collection[Any],
                                                         cognite_client: CogniteClient = 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: SequenceType[int] = None, asset_subtree_ids: SequenceType[Dict[str, Any]] = None, created_time: Union[Dict[str, Any], TimestampRange] = None, last_updated_time: Union[Dict[str, Any], TimestampRange] = None, data_set_ids: SequenceType[Dict[str, Any]] = None, cognite_client: CogniteClient = 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** (*SequenceType[int]*) – Return only sequences linked to one of the specified assets.
- **asset_subtree_ids** (*SequenceType[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** (*SequenceType[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:      Col-
                                                    lection[Any],      cog-
                                                    nite_client:    Cognite-
                                                    Client = None)
```

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

```
class cognite.client.data_classes.sequences.SequenceUpdate (id: Optional[int] =
                                                    None, external_id:
                                                    Optional[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.

2.4.13 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, Sequence[str]]*, *recursive: bool = False*) → None

Delete one or more databases.

Parameters

- **name** (*Union[str, Sequence[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, Sequence[str]]*) → `None`

Delete one or more tables.

Parameters

- **db_name** (*str*) – Database to delete tables from.
- **name** (*Union[str, Sequence[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 table '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: Optional[int] = None, max_last_updated_time: Optional[int] = None, columns: Optional[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[Sequence[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[Sequence[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"}}
>>> c.raw.rows.insert("db1", "table1", rows)
```

Delete rows from a table

`RawRowsAPI.delete` (*db_name: str, table_name: str, key: Union[str, Sequence[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, Sequence[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)
```

Retrieve pandas dataframe

`RawRowsAPI.retrieve_dataframe` (*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*) → `pandas.DataFrame`

Retrieve rows in a table as a pandas dataframe.

Rowkeys are used as the index.

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 in a pandas dataframe.

Return type pandas.DataFrame

Examples

Get dataframe:

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

Insert pandas dataframe

RawRowsAPI.**insert_dataframe** (*db_name: str, table_name: str, dataframe: Any, ensure_parent: bool = False*) → None

Insert pandas dataframe into a table

Use index as rowkeys.

Parameters

- **db_name** (*str*) – Name of the database.
- **table_name** (*str*) – Name of the table.
- **dataframe** (*pandas.DataFrame*) – The dataframe to insert. Index will be used as rowkeys.
- **ensure_parent** (*bool*) – Create database/table if they don't already exist.

Returns None

Examples

Insert new rows into a table:

```
>>> import pandas as pd
>>> from cognite.client import CogniteClient
>>>
>>> c = CogniteClient()
>>> df = pd.DataFrame(data={"a": 1, "b": 2}, index=["r1", "r2", "r3"])
>>> res = c.raw.rows.insert_dataframe("db1", "table1", df)
```

Data classes

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

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

A NoSQL database to store customer data.

Parameters

- **name** (*str*) – Unique name of a database.
- **created_time** (*int*) – Time the database was created.
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

tables (*limit: Optional[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: Collection[Any],  
cognite_client: CogniteClient =  
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, cog-  
nite_client: CogniteClient = 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 () → *pandas.DataFrame*

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: Collection[Any], cog-  
nite_client: CogniteClient = None)
```

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

to_pandas () → *pandas.DataFrame*

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, created_time: int = None,  
cognite_client: CogniteClient = None)
```

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

A NoSQL database table to store customer data

Parameters

- **name** (*str*) – Unique name of the table
- **created_time** (*int*) – Time the table was created.

- **cognite_client** (*CogniteClient*) – The client to associate with this object.

rows (*key: Optional[str] = None, limit: Optional[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: Collection[Any], cognite_client: CogniteClient = None*)
Bases: *cognite.client.data_classes._base.CogniteResourceList*

2.4.14 Relationships

Retrieve a relationship by external id

RelationshipsAPI.**retrieve** (*external_id: str, fetch_resources: bool = False*) → Optional[cognite.client.data_classes.relationships.Relationship]
Retrieve a single relationship by external id.

Parameters

- **external_id** (*str*) – External ID
- **fetch_resources** (*bool*) – if true, will try to return the full resources referenced by the relationship in the source and target fields.

Returns Requested relationship or None if it does not exist.

Return type Optional[*Relationship*]

Examples

Get relationship by external id:

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

Retrieve multiple relationships by external id

RelationshipsAPI.**retrieve_multiple** (*external_ids: Sequence[str], fetch_resources: bool = False*) → cognite.client.data_classes.relationships.RelationshipList
Retrieve multiple relationships by external id.

Parameters

- **external_ids** (*Sequence[str]*) – External IDs
- **fetch_resources** (*bool*) – if true, will try to return the full resources referenced by the relationship in the source and target fields.

Returns The requested relationships.

Return type *RelationshipList*

Examples

Get relationships by external id:

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

List relationships

RelationshipsAPI.**list** (*source_external_ids: Optional[Sequence[str]] = None, source_types: Optional[Sequence[str]] = None, target_external_ids: Optional[Sequence[str]] = None, target_types: Optional[Sequence[str]] = None, data_set_ids: Union[int, Sequence[int], None] = None, data_set_external_ids: Union[str, Sequence[str], None] = None, start_time: Optional[Dict[str, int]] = None, end_time: Optional[Dict[str, int]] = None, confidence: Optional[Dict[str, int]] = None, last_updated_time: Optional[Dict[str, int]] = None, created_time: Optional[Dict[str, int]] = None, active_at_time: Optional[Dict[str, int]] = None, labels: Optional[cognite.client.data_classes.labels.LabelFilter] = None, limit: int = 100, partitions: Optional[int] = None, fetch_resources: bool = False*) → *cognite.client.data_classes.relationships.RelationshipList*

Lists relationships stored in the project based on a query filter given in the payload of this request. Up to 1000 relationships can be retrieved in one operation.

Parameters

- **source_external_ids** (*Sequence[str]*) – Include relationships that have any of these values in their source External Id field
- **source_types** (*Sequence[str]*) – Include relationships that have any of these values in their source Type field
- **target_external_ids** (*Sequence[str]*) – Include relationships that have any of these values in their target External Id field
- **target_types** (*Sequence[str]*) – Include relationships that have any of these values in their target Type field
- **data_set_ids** (*Union[int, Sequence[int]]*) – Return only relationships in the specified data set(s) with this id / these ids.
- **data_set_external_ids** (*Union[str, Sequence[str]]*) – Return only relationships in the specified data set(s) with this external id / these external ids.
- **start_time** (*Dict[str, int]*) – Range between two timestamps, minimum and maximum milli seconds (inclusive)
- **end_time** (*Dict[str, int]*) – Range between two timestamps, minimum and maximum milli seconds (inclusive)
- **confidence** (*Dict[str, int]*) – Range to filter the field for (inclusive).
- **last_updated_time** (*Dict[str, Any]*) – Range to filter the field for (inclusive).
- **created_time** (*Dict[str, int]*) – Range to filter the field for (inclusive).

- **active_at_time** (*Dict[str, int]*) – Limits results to those active at any point within the given time range, i.e. if there is any overlap in the intervals [activeAtTime.min, activeAtTime.max] and [startTime, endTime], where both intervals are inclusive. If a relationship does not have a startTime, it is regarded as active from the beginning of time by this filter. If it does not have an endTime it will be regarded as active until the end of time. Similarly, if a min is not supplied to the filter, the min will be implicitly set to the beginning of time, and if a max is not supplied, the max will be implicitly set to the end of time.
- **labels** (*LabelFilter*) – Return only the resource matching the specified label constraints.
- **limit** (*int*) – Maximum number of relationships to return. Defaults to 100. Set to -1, float(“inf”) or None to return all items.
- **partitions** (*int*) – Retrieve relationships in parallel using this number of workers. Also requires *limit=None* to be passed.
- **fetch_resources** (*bool*) – if true, will try to return the full resources referenced by the relationship in the source and target fields.

Returns List of requested relationships

Return type *RelationshipList*

Examples

List relationships:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> relationship_list = c.relationships.list(limit=5)
```

Iterate over relationships:

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

Create a relationship

`RelationshipsAPI.create` (*relationship: Union[cognite.client.data_classes.relationships.Relationship, Sequence[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, Sequence[Relationship]]*) – Relationship or list of relationships to create.

Returns Created relationship(s)

Return type Union[*Relationship, RelationshipList*]

Note:

- The `source_type` and `target_type` field in the `Relationship(s)` can be any string among “Asset”, “Time-Series”, “File”, “Event”, “Sequence”;

- Do not provide the value for the source and target arguments of the Relationship class, only source_external_id / source_type and target_external_id / target_type. These (source and target) are used as part of fetching actual resources specified in other fields.

Examples

Create a new relationship specifying object type and external id for source and target:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import Relationship
>>> c = CogniteClient()
>>> flowrel1 = Relationship(
...     external_id="flow_1",
...     source_external_id="source_ext_id",
...     source_type="asset",
...     target_external_id="target_ext_id",
...     target_type="event",
...     confidence=0.1,
...     data_set_id=1234
... )
>>> flowrel2 = Relationship(
...     external_id="flow_2",
...     source_external_id="source_ext_id",
...     source_type="asset",
...     target_external_id="target_ext_id",
...     target_type="event",
...     confidence=0.1,
...     data_set_id=1234
... )
>>> res = c.relationships.create([flowrel1, flowrel2])
```

Update relationships

RelationshipsAPI.**update** (*item*: *Union[cognite.client.data_classes.relationships.Relationship, cognite.client.data_classes.relationships.RelationshipUpdate, Sequence[Union[cognite.client.data_classes.relationships.Relationship, cognite.client.data_classes.relationships.RelationshipUpdate]]]*) → *Union[cognite.client.data_classes.relationships.Relationship, cognite.client.data_classes.relationships.RelationshipList]*

Update one or more relationships Currently, a full replacement of labels on a relationship is not supported (only partial add/remove updates). See the example below on how to perform partial labels update.

Parameters *item* (*Union[Relationship, RelationshipUpdate, Sequence[Union[Relationship, RelationshipUpdate]]]*) – Relationship(s) to update

Returns Updated relationship(s)

Return type *Union[Relationship, RelationshipList]*

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()
>>> rel = c.relationships.retrieve(external_id="flow1")
>>> rel.confidence = 0.75
>>> res = c.relationships.update(rel)
```

Perform a partial update on a relationship, setting a source_external_id and a confidence:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import RelationshipUpdate
>>> c = CogniteClient()
>>> my_update = RelationshipUpdate(external_id="flow_1").source_external_id.set(
↳ "alternate_source").confidence.set(0.97)
>>> res1 = c.relationships.update(my_update)
>>> # Remove an already set optional field like so
>>> another_update = RelationshipUpdate(external_id="flow_1").confidence.set(None)
>>> res2 = c.relationships.update(another_update)
```

Attach labels to a relationship:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import RelationshipUpdate
>>> c = CogniteClient()
>>> my_update = RelationshipUpdate(external_id="flow_1").labels.add(["PUMP",
↳ "VERIFIED"])
>>> res = c.relationships.update(my_update)
```

Detach a single label from a relationship:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import RelationshipUpdate
>>> c = CogniteClient()
>>> my_update = RelationshipUpdate(external_id="flow_1").labels.remove("PUMP")
>>> res = c.relationships.update(my_update)
```

Delete relationships

RelationshipsAPI.**delete**(*external_id: Union[str, Sequence[str]]*, *ignore_unknown_ids: bool = False*) → None
Delete one or more relationships.

Parameters

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

Returns None

Examples

Delete relationships by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.relationships.delete(external_id=["a", "b"])
```

Data classes

```
class cognite.client.data_classes.relationships.Relationship(external_id:
    str = None,
    source_external_id:
    str = None,
    source_type: str
    = None, source:
    Union[Asset,
    TimeSeries,
    FileMetadata,
    Sequence, Event,
    Dict] = None,
    target_external_id:
    str = None, tar-
    get_type: str =
    None, target:
    Union[Asset,
    TimeSeries,
    FileMetadata,
    Sequence, Event,
    Dict] = None,
    start_time: int =
    None, end_time:
    int = None, con-
    fidence: float =
    None, data_set_id:
    int = None,
    labels: Sequence-
    Type[Union[Label,
    str, LabelDefini-
    tion, dict]] = None,
    created_time:
    int = None,
    last_updated_time:
    int = None,
    cognite_client:
    CogniteClient =
    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

- **external_id** (*str*) – External id of the relationship, must be unique within the project.
- **source_external_id** (*str*) – External id of the CDF resource that constitutes the relationship source.
- **source_type** (*str*) – The CDF resource type of the relationship source. Must be one of the specified values.
- **source** (`Union[Asset, TimeSeries, FileMetadata, Event, Sequence, Dict]`) – The full resource referenced by the `source_external_id` and `source_type` fields.

- **target_external_id** (*str*) – External id of the CDF resource that constitutes the relationship target.
- **target_type** (*str*) – The CDF resource type of the relationship target. Must be one of the specified values.
- **target** (*Union[Asset, TimeSeries, FileMetadata, Event, Sequence, Dict]*) – The full resource referenced by the `target_external_id` and `target_type` fields.
- **start_time** (*int*) – Time, in milliseconds since Jan. 1, 1970, when the 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 the 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. Generated relationships should provide a realistic score on the likelihood of the existence of the relationship. Relationships without a confidence value can be interpreted at the discretion of each project.
- **data_set_id** (*int*) – The id of the dataset this relationship belongs to.
- **labels** (*SequenceType[Label]*) – A list of the labels associated with this resource item.
- **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 (source_external_ids:
    Sequence-
    Type[str]
    = None,
    source_types:
    Sequence-
    Type[str]
    = None,
    tar-
    get_external_ids:
    Sequence-
    Type[str]
    = None,
    tar-
    get_types:
    Sequence-
    Type[str]
    = None,
    data_set_ids:
    Sequence-
    Type[Dict[str,
    Any]] =
    None,
    start_time:
    Dict[str,
    int] =
    None,
    end_time:
    Dict[str,
    int] =
    None,
    confi-
    dence:
    Dict[str,
    int]
    = None,
    last_updated_time:
    Dict[str,
    int] =
    None, created_time:
    Dict[str,
    int] =
    None, ac-
    tive_at_time:
    Dict[str,
    int] =
    None,
    labels:
    LabelFil-
    ter =
    None,
    cog-
    nite_client:
    Cognite-
    Client =
```

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

Filter on relationships with exact match. Multiple filter elements in one property, e.g. `sourceExternalIds: ["a", "b"]`, will return all relationships where the `sourceExternalId` field is either `a` or `b`. Filters in multiple properties will return the relationships that match all criteria. If the filter is not specified it default to an empty filter.

Parameters

- **source_external_ids** (`Sequence[str]`) – Include relationships that have any of these values in their `sourceExternalId` field
- **source_types** (`Sequence[str]`) – Include relationships that have any of these values in their `sourceType` field
- **target_external_ids** (`Sequence[str]`) – Include relationships that have any of these values in their `targetExternalId` field
- **target_types** (`Sequence[str]`) – Include relationships that have any of these values in their `targetType` field
- **data_set_ids** (`Sequence[Dict[str, Any]]`) – Either one of `internalId` (int) or `externalId` (str)
- **start_time** (`Dict[str, int]`) – Range between two timestamps, minimum and maximum milliseconds (inclusive)
- **end_time** (`Dict[str, int]`) – Range between two timestamps, minimum and maximum milliseconds (inclusive)
- **confidence** (`Dict[str, int]`) – Range to filter the field for (inclusive).
- **last_updated_time** (`Dict[str, Any]`) – Range to filter the field for (inclusive).
- **created_time** (`Dict[str, int]`) – Range to filter the field for (inclusive).
- **active_at_time** (`Dict[str, int]`) – Limits results to those active at any point within the given time range, i.e. if there is any overlap in the intervals [`activeAtTime.min`, `activeAtTime.max`] and [`startTime`, `endTime`], where both intervals are inclusive. If a relationship does not have a `startTime`, it is regarded as active from the beginning of time by this filter. If it does not have an `endTime` it will be regarded as active until the end of time. Similarly, if a `min` is not supplied to the filter, the `min` will be implicitly set to the beginning of time, and if a `max` is not supplied, the `max` will be implicitly set to the end of time.
- **labels** (`LabelFilter`) – Return only the resource matching the specified label constraints.
- **cognite_client** (`CogniteClient`) – The client to associate with this object.

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]`

```
class cognite.client.data_classes.relationships.RelationshipList (resources:
                                                    Collec-
                                                    tion[Any],
                                                    cog-
                                                    nite_client:
                                                    Cognite-
                                                    Client      =
                                                    None)

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

```
class cognite.client.data_classes.relationships.RelationshipUpdate (external_id:
                                                                    str)
```

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

Update applied to a single relationship

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

2.4.15 Geospatial

Note: Check <https://github.com/cognitedata/geospatial-examples> for some complete examples.

Create feature types

```
GeospatialAPI.create_feature_types (feature_type: Union[cognite.client.data_classes.geospatial.FeatureType,
                                                    Sequence[cognite.client.data_classes.geospatial.FeatureType]])
→ Union[cognite.client.data_classes.geospatial.FeatureType,
cognite.client.data_classes.geospatial.FeatureTypeList]
```

Creates feature types <<https://docs.cognite.com/api/v1/#operation/createFeatureTypes>>

Parameters **feature_type** (*Union[FeatureType, Sequence[FeatureType]]*) – feature type definition or list of feature type definitions to create.

Returns Created feature type definition(s)

Return type Union[*FeatureType, FeatureTypeList*]

Examples

Create new type definitions:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes.geospatial import FeatureType
>>> c = CogniteClient()
>>> feature_types = [
...     FeatureType(external_id="wells", properties={"location": {"type": "POINT",
↪ "srid": 4326}})
...     FeatureType(
...         external_id="cities",
...         properties={"name": {"type": "STRING", "size": 10}},
...         search_spec={"name_index": {"properties": ["name"]}}
...     )
... ]
```

(continues on next page)

(continued from previous page)

```
... ]
>>> res = c.geospatial.create_feature_types(feature_types)
```

Delete feature types

`GeospatialAPI.delete_feature_types` (*external_id*: `Union[str, Sequence[str]]`, *recursive*: `bool = False`) → `None`
Delete one or more feature type <<https://docs.cognite.com/api/v1/#operation/GeospatialDeleteFeatureTypes>>

Parameters

- **external_id** (`Union[str, Sequence[str]]`) – External ID or list of external ids
- **recursive** (`bool`) – if `true` the features will also be dropped

Returns `None`

Examples

Delete feature type definitions external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.geospatial.delete_feature_types(external_id=["wells", "cities"])
```

List feature types

`GeospatialAPI.list_feature_types` () → `cognite.client.data_classes.geospatial.FeatureTypeList`
List feature types <<https://docs.cognite.com/api/v1/#operation/listFeatureTypes>>

Returns List of feature types

Return type `FeatureTypeList`

Examples

Iterate over feature type definitions:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> for feature_type in c.geospatial.list_feature_types():
...     feature_type # do something with the feature type definition
```

Retrieve feature types

`GeospatialAPI.retrieve_feature_types` (*external_id*: `Union[str, List[str]]`) → `Union[cognite.client.data_classes.geospatial.FeatureType, cognite.client.data_classes.geospatial.FeatureTypeList]`
Retrieve feature types <<https://docs.cognite.com/api/v1/#operation/getFeatureTypesByIds>>

Parameters **external_id** (`Union[str, List[str]]`) – External ID

Returns Requested Type or `None` if it does not exist.

Return type *FeatureTypeList*

Examples

Get Type by external id:

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

Update feature types

`GeospatialAPI.patch_feature_types` (*patch*: `Union[cognite.client.data_classes.geospatial.FeatureTypePatch, Sequence[cognite.client.data_classes.geospatial.FeatureTypePatch]]`)
→ `cognite.client.data_classes.geospatial.FeatureTypeList`

Patch feature types <<https://docs.cognite.com/api/v1/#operation/updateFeatureTypes>>

Parameters `patch` (`Union[FeatureTypePatch, Sequence[FeatureTypePatch]]`)
– the patch to apply

Returns The patched feature types.

Return type *FeatureTypeList*

Examples

Add one property to a feature type and add indexes

```
>>> from cognite.client.data_classes.geospatial import Patches
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.geospatial.patch_feature_types(
...     patch=FeatureTypePatch(
...         external_id="wells",
...         property_patches=Patches(add={"altitude": {"type": "DOUBLE"}}),
...         search_spec_patches=Patches(
...             add={
...                 "altitude_idx": {"properties": ["altitude"]},
...                 "composite_idx": {"properties": ["location", "altitude"]}
...             }
...         )
...     )
... )
```

Add an additional index to an existing property

```
>>> from cognite.client.data_classes.geospatial import Patches
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.geospatial.patch_feature_types(
...     patch=FeatureTypePatch(
...         external_id="wells",
...         search_spec_patches=Patches(add={"location_idx": {"properties": [
...             ↪"location"]}})
...     )
... )
```

Create features

`GeospatialAPI.create_features` (*feature_type_external_id*: *str*, *feature*: *Union[cognite.client.data_classes.geospatial.Feature, Sequence[cognite.client.data_classes.geospatial.Feature], cognite.client.data_classes.geospatial.FeatureList]*, *allow_crs_transformation*: *bool = False*, *chunk_size*: *Optional[int] = None*) → *Union[cognite.client.data_classes.geospatial.Feature, cognite.client.data_classes.geospatial.FeatureList]*

Creates features <<https://docs.cognite.com/api/v1/#operation/createFeatures>>

Parameters

- **feature_type_external_id** – Feature type definition for the features to create.
- **feature** – one feature or a list of features to create or a `FeatureList` object
- **allow_crs_transformation** – If true, then input geometries will be transformed into the Coordinate Reference System defined in the feature type specification. When it is false, then requests with geometries in Coordinate Reference System different from the ones defined in the feature type will result in `CogniteAPIError` exception.
- **chunk_size** – maximum number of items in a single request to the api

Returns Created features

Return type `Union[Feature, FeatureList]`

Examples

Create a new feature type and corresponding feature:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> feature_types = [
...     FeatureType(
...         external_id="my_feature_type",
...         properties={
...             "location": {"type": "POINT", "srid": 4326},
...             "temperature": {"type": "DOUBLE"}
...         }
...     )
... ]
>>> res = c.geospatial.create_feature_types(feature_types)
>>> res = c.geospatial.create_features(
...     feature_type_external_id="my_feature_type",
...     feature=Feature(
...         external_id="my_feature",
...         location={"wkt": "POINT(1 1)"},
...         temperature=12.4
...     )
... )
```

Delete features

`GeospatialAPI.delete_features` (*feature_type_external_id*: *str*, *external_id*: *Union[str, Sequence[str], None] = None*) → *None*

Delete one or more feature <<https://docs.cognite.com/api/v1/#operation/deleteFeatures>>

Parameters

- **feature_type_external_id** – Feature type external id for the features to delete.
- **external_id** (*Union[str, Sequence[str]]*) – External ID or list of external ids

Returns None**Examples**

Delete feature type definitions external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.geospatial.delete_features(
...     feature_type_external_id="my_feature_type",
...     external_id=my_feature
... )
```

Retrieve features

`GeospatialAPI.retrieve_features` (*feature_type_external_id: str, external_id: Union[str, List[str]], properties: Optional[Dict[str, Any]] = None*)
→ `Union[cognite.client.data_classes.geospatial.FeatureList, cognite.client.data_classes.geospatial.Feature]`

Retrieve features <<https://docs.cognite.com/api/v1/#operation/getFeaturesByIds>>

Parameters

- **feature_type_external_id** – Feature type external id for the features to retrieve.
- **external_id** (*Union[str, List[str]]*) – External ID or list of external ids
- **properties** (*Dict[str, Any]*) – the output property selection

Returns Requested features or None if it does not exist.**Return type** *FeatureList***Examples**

Retrieve one feature by its external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.geospatial.retrieve_features(
...     feature_type_external_id="my_feature_type",
...     external_id="my_feature"
... )
```

Update features

`GeospatialAPI.update_features` (*feature_type_external_id*: *str*, *feature*: *Union[cognite.client.data_classes.geospatial.Feature, Sequence[cognite.client.data_classes.geospatial.Feature]]*, *allow_crs_transformation*: *bool = False*, *chunk_size*: *Optional[int] = None*) → `cognite.client.data_classes.geospatial.FeatureList`

Update features <<https://docs.cognite.com/api/v1/#operation/updateFeatures>>

Parameters

- **feature_type_external_id** – Feature type definition for the features to update.
- **feature** (*Union[Feature, Sequence[Feature]]*) – feature or list of features.
- **allow_crs_transformation** – If true, then input geometries will be transformed into the Coordinate Reference System defined in the feature type specification. When it is false, then requests with geometries in Coordinate Reference System different from the ones defined in the feature type will result in `CogniteAPIError` exception.
- **chunk_size** – maximum number of items in a single request to the api

Returns Updated features

Return type *FeatureList*

Examples

Update one feature:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> my_feature = c.geospatial.create_features(
...     feature_type_external_id="my_feature_type",
...     feature=Feature(external_id="my_feature", temperature=12.4)
... )
>>> my_updated_feature = c.geospatial.update_features(
...     feature_type_external_id="my_feature_type",
...     feature=Feature(external_id="my_feature", temperature=6.237)
... )
```

List features

`GeospatialAPI.list_features` (*feature_type_external_id*: *str*, *filter*: *Optional[Dict[str, Any]] = None*, *properties*: *Optional[Dict[str, Any]] = None*, *limit*: *int = 100*, *allow_crs_transformation*: *bool = False*) → `cognite.client.data_classes.geospatial.FeatureList`

List features <<https://docs.cognite.com/api/v1/#operation/listFeatures>>

This method allows to filter all features.

Parameters

- **feature_type_external_id** – the feature type to list features for
- **filter** (*Dict[str, Any]*) – the list filter
- **limit** (*int, optional*) – Maximum number of features to return. Defaults to 25. Set to -1, float("inf") or None to return all features.

- **properties** (*Dict[str, Any]*) – the output property selection
- **allow_crs_transformation** – If true, then input geometries if existing in the filter will be transformed into the Coordinate Reference System defined in the feature type specification. When it is false, then requests with geometries in Coordinate Reference System different from the ones defined in the feature type will result in `CogniteAPIError` exception.

Returns The filtered features

Return type *FeatureList*

Examples

List features:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> my_feature_type = c.geospatial.retrieve_feature_types(
...     external_id="my_feature_type"
... )
>>> my_feature = c.geospatial.create_features(
...     feature_type_external_id=my_feature_type,
...     feature=Feature(
...         external_id="my_feature",
...         temperature=12.4,
...         location={"wkt": "POINT(0 1)"}
...     )
... )
>>> res = c.geospatial.list_features(
...     feature_type_external_id="my_feature_type",
...     filter={"range": {"property": "temperature", "gt": 12.0}}
... )
>>> for f in res:
...     # do something with the features
```

Search for features and select output properties:

```
>>> res = c.geospatial.list_features(
...     feature_type_external_id=my_feature_type,
...     filter={},
...     properties={"temperature": {}, "pressure": {}}
... )
```

Search for features with spatial filters:

```
>>> res = c.geospatial.list_features(
...     feature_type_external_id=my_feature_type,
...     filter={"stWithin": {
...         "property": "location",
...         "value": {"wkt": "POLYGON((0 0, 0 1, 1 1, 0 0))"}
...     }}
... )
```

Search features

`GeospatialAPI.search_features` (*feature_type_external_id*: *str*, *filter*: *Optional[Dict[str, Any]] = None*, *properties*: *Optional[Dict[str, Any]] = None*, *limit*: *int = 100*, *order_by*: *Optional[Sequence[cognite.client.data_classes.geospatial.OrderSpec]] = None*, *allow_crs_transformation*: *bool = False*) → `cognite.client.data_classes.geospatial.FeatureList`

Search for features <<https://docs.cognite.com/api/v1/#operation/searchFeatures>>

This method allows to order the result by one or more of the properties of the feature type. However, the number of items returned is limited to 1000 and there is no support for cursors yet. If you need to return more than 1000 items, use the `stream_features(...)` method instead.

Parameters

- **feature_type_external_id** – the feature type to search for
- **filter** (*Dict[str, Any]*) – the search filter
- **limit** (*int*) – maximum number of results
- **properties** (*Dict[str, Any]*) – the output property selection
- **order_by** (*Sequence[OrderSpec]*) – the order specification
- **allow_crs_transformation** – If true, then input geometries will be transformed into the Coordinate Reference System defined in the feature type specification. When it is false, then requests with geometries in Coordinate Reference System different from the ones defined in the feature type will result in `CogniteAPIError` exception.

Returns the filtered features

Return type *FeatureList*

Examples

Search for features:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> my_feature_type = c.geospatial.retrieve_feature_types(
...     external_id="my_feature_type"
... )
>>> my_feature = c.geospatial.create_features(
...     feature_type_external_id=my_feature_type,
...     feature=Feature(
...         external_id="my_feature",
...         temperature=12.4,
...         location={"wkt": "POINT(0 1)"}
...     )
... )
>>> res = c.geospatial.search_features(
...     feature_type_external_id="my_feature_type",
...     filter={"range": {"property": "temperature", "gt": 12.0}}
... )
>>> for f in res:
...     # do something with the features
```

Search for features and select output properties:

```
>>> res = c.geospatial.search_features(  
...     feature_type_external_id=my_feature_type,  
...     filter={},  
...     properties={"temperature": {}, "pressure": {}}  
... )
```

Search for features and order results:

```
>>> res = c.geospatial.search_features(  
...     feature_type_external_id=my_feature_type,  
...     filter={},  
...     order_by=[  
...         OrderSpec("temperature", "ASC"),  
...         OrderSpec("pressure", "DESC")  
...     ]  
... )
```

Search for features with spatial filters:

```
>>> res = c.geospatial.search_features(  
...     feature_type_external_id=my_feature_type,  
...     filter={"stWithin": {  
...         "property": "location",  
...         "value": {"wkt": "POLYGON((0 0, 0 1, 1 1, 0 0))"}  
...     }}  
... )
```

Combining multiple filters:

```
>>> res = c.geospatial.search_features(  
...     feature_type_external_id=my_feature_type,  
...     filter={"and": [  
...         {"range": {"property": "temperature", "gt": 12.0}},  
...         {"stWithin": {  
...             "property": "location",  
...             "value": {"wkt": "POLYGON((0 0, 0 1, 1 1, 0 0))"}  
...         }}  
...     ]}  
... )
```

```
>>> res = c.geospatial.search_features(  
...     feature_type_external_id=my_feature_type,  
...     filter={"or": [  
...         {"range": {"property": "temperature", "gt": 12.0}},  
...         {"stWithin": {  
...             "property": "location",  
...             "value": {"wkt": "POLYGON((0 0, 0 1, 1 1, 0 0))"}  
...         }}  
...     ]}  
... )
```

Stream features

`GeospatialAPI.stream_features` (*feature_type_external_id*: *str*, *filter*: *Optional[Dict[str, Any]] = None*, *properties*: *Optional[Dict[str, Any]] = None*, *allow_crs_transformation*: *bool = False*) → `Generator[cognite.client.data_classes.geospatial.Feature, None, None]`

Stream features <<https://docs.cognite.com/api/v1/#operation/searchFeaturesStreaming>>

This method allows to return any number of items until the underlying api calls times out. The order of the result items is not deterministic. If you need to order the results, use the `search_features(...)` method instead.

Parameters

- **feature_type_external_id** – the feature type to search for
- **filter** (*Dict[str, Any]*) – the search filter
- **properties** (*Dict[str, Any]*) – the output property selection
- **allow_crs_transformation** – If true, then input geometries will be transformed into the Coordinate Reference System defined in the feature type specification. When it is false, then requests with geometries in Coordinate Reference System different from the ones defined in the feature type will result in `CogniteAPIError` exception.

Returns a generator for the filtered features

Return type `Generator[Feature]`

Examples

Stream features:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> my_feature = c.geospatial.create_features(
...     feature_type_external_id="my_feature_type",
...     feature=Feature(external_id="my_feature", temperature=12.4)
... )
>>> features = c.geospatial.stream_features(
...     feature_type_external_id="my_feature_type",
...     filter={"range": {"property": "temperature", "gt": 12.0}}
... )
>>> for f in features:
...     # do something with the features
```

Stream features and select output properties:

```
>>> features = c.geospatial.stream_features(
...     feature_type_external_id="my_feature_type",
...     filter={},
...     properties={"temperature": {}, "pressure": {}}
... )
>>> for f in features:
...     # do something with the features
```

Aggregate features

`GeospatialAPI.aggregate_features` (*feature_type_external_id: str, property: Optional[str] = None, aggregates: Optional[Sequence[str]] = None, filter: Optional[Dict[str, Any]] = None, group_by: Optional[Sequence[str]] = None, order_by: Optional[Sequence[cognite.client.data_classes.geospatial.OrderSpec]] = None, output: Optional[Dict[str, Any]] = None*) → `cognite.client.data_classes.geospatial.FeatureAggregateList`
 Aggregate filtered features <<https://docs.cognite.com/api/v1/#operation/aggregateFeatures>>

Parameters

- **feature_type_external_id** – the feature type to filter features from
- **filter** (*Dict[str, Any]*) – the search filter
- **property** (*str*) – the property for which aggregates should be calculated
- **aggregates** (*Sequence[str]*) – list of aggregates to be calculated
- **group_by** (*Sequence[str]*) – list of properties to group by with
- **order_by** (*Sequence[OrderSpec]*) – the order specification
- **output** (*Dict[str, Any]*) – the aggregate output

Returns the filtered features

Return type *FeatureAggregateList*

Examples

Aggregate property of features:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> my_feature = c.geospatial.create_features(
...     feature_type_external_id="my_feature_type",
...     feature=Feature(external_id="my_feature", temperature=12.4)
... )
>>> res_deprecated = c.geospatial.aggregate_features(
...     feature_type_external_id="my_feature_type",
...     filter={"range": {"property": "temperature", "gt": 12.0}},
...     property="temperature",
...     aggregates=["max", "min"],
...     group_by=["category"],
...     order_by=[OrderSpec("category", "ASC")]
... ) # deprecated
>>> res = c.geospatial.aggregate_features(
...     feature_type_external_id="my_feature_type",
...     filter={"range": {"property": "temperature", "gt": 12.0}},
...     group_by=["category"],
...     order_by=[OrderSpec("category", "ASC")],
...     output={"min_temperature": {"min": {"property": "temperature"}},
...            "max_volume": {"max": {"property": "volume"}}
...     }
... )
>>> for a in res:
...     # loop over aggregates in different groups
```

Get coordinate reference systems

GeospatialAPI.**get_coordinate_reference_systems** (*srids*: Union[int, Sequence[int]]) → cog-nite.client.data_classes.geospatial.CoordinateReferenceSystemList
 Get Coordinate Reference Systems
[getCoordinateReferenceSystem](https://docs.cognite.com/api/v1/#operation/getCoordinateReferenceSystem)

Parameters *srids* – (Union[int, Sequence[int]]): SRID or list of SRIDs

Returns Requested CRSs.

Return type *CoordinateReferenceSystemList*

Examples

Get two CRS definitions:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> crs = c.geospatial.get_coordinate_reference_systems(srids=[4326, 4327])
```

List coordinate reference systems

GeospatialAPI.**list_coordinate_reference_systems** (*only_custom*: bool = False) → cog-nite.client.data_classes.geospatial.CoordinateReferenceSystemList
 List Coordinate Reference Systems
[listGeospatialCoordinateReferenceSystems](https://docs.cognite.com/api/v1/#operation/listGeospatialCoordinateReferenceSystems)

Parameters *only_custom* (bool) – list only custom CRSs or not

Returns list of CRSs.

Return type *CoordinateReferenceSystemList*

Examples

Fetch all custom CRSs:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> crs = c.geospatial.list_coordinate_reference_systems(only_custom=True)
```

Create coordinate reference systems

GeospatialAPI.**create_coordinate_reference_systems** (*crs*: Union[cognite.client.data_classes.geospatial.CoordinateReferenceSystem, Sequence[cognite.client.data_classes.geospatial.CoordinateReferenceSystem]]) → cog-nite.client.data_classes.geospatial.CoordinateReferenceSystemList
 Create Coordinate Reference System
[createGeospatialCoordinateReferenceSystems](https://docs.cognite.com/api/v1/#operation/createGeospatialCoordinateReferenceSystems)

Parameters *crs* – a CoordinateReferenceSystem or a list of CoordinateReferenceSystem

Returns list of CRSs.

Return type *CoordinateReferenceSystemList*

Examples

Create a custom CRS:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> custom_crs = CoordinateReferenceSystem(
...     srid = 121111,
...     wkt=(
...         'PROJCS["NTF (Paris) / Lambert zone II",'
...         ' GEOGCS["NTF (Paris)",'
...         ' DATUM["Nouvelle_Triangulation_Francaise_Paris",'
...         ' SPHEROID["Clarke 1880 (IGN)",6378249.2,293.4660212936265,'
...         ' AUTHORITY["EPSG","7011"]],'
...         ' TOWGS84[-168,-60,320,0,0,0,0],'
...         ' AUTHORITY["EPSG","6807"]],'
...         ' PRIMEM["Paris",2.33722917,'
...         ' AUTHORITY["EPSG","8903"]],'
...         ' UNIT["grad",0.01570796326794897,'
...         ' AUTHORITY["EPSG","9105"]],'
...         ' AUTHORITY["EPSG","4807"]],'
...         ' PROJECTION["Lambert_Conformal_Conic_1SP"],'
...         ' PARAMETER["latitude_of_origin",52],'
...         ' PARAMETER["central_meridian",0],'
...         ' PARAMETER["scale_factor",0.99987742],'
...         ' PARAMETER["false_easting",600000],'
...         ' PARAMETER["false_northing",2200000],'
...         ' UNIT["metre",1,'
...         ' AUTHORITY["EPSG","9001"]],'
...         ' AXIS["X",EAST],'
...         ' AXIS["Y",NORTH],'
...         ' AUTHORITY["EPSG","27572"]]'
...     ),
...     proj_string=(
...         '+proj=lcc +lat_1=46.8 +lat_0=46.8 +lon_0=0 +k_0=0.99987742 '
...         '+x_0=600000 +y_0=2200000 +a=6378249.2 +b=6356515 '
...         '+towgs84=-168,-60,320,0,0,0,0 +pm=paris +units=m +no_defs'
...     )
... )
>>> crs = c.geospatial.create_coordinate_reference_systems(custom_crs)
```

Put raster data

`GeospatialAPI.put_raster` (*feature_type_external_id*: *str*, *feature_external_id*: *str*, *raster_property_name*: *str*, *raster_format*: *str*, *raster_srid*: *int*, *file*: *str*, *allow_crs_transformation*: *bool* = *False*, *raster_scale_x*: *Optional[float]* = *None*, *raster_scale_y*: *Optional[float]* = *None*) → `cognite.client.data_classes.geospatial.RasterMetadata`

Put raster <<https://docs.cognite.com/api/v1/#tag/Geospatial/operation/putRaster>>

Parameters

- **feature_type_external_id** – Feature type definition for the features to create.
- **feature_external_id** – one feature or a list of features to create

- **raster_property_name** – the raster property name
- **raster_format** – the raster input format
- **raster_srid** – the associated SRID for the raster
- **file** – the path to the file of the raster
- **allow_crs_transformation** – When the parameter is false, requests with rasters in Coordinate Reference System different from the one defined in the feature type will result in bad request response code.
- **raster_scale_x** – the X component of the pixel width in units of coordinate reference system
- **raster_scale_y** – the Y component of the pixel height in units of coordinate reference system

Returns the raster metadata if it was ingested successfully

Return type *RasterMetadata*

Examples

Put a raster in a feature raster property:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> feature_type = ...
>>> feature = ...
>>> raster_property_name = ...
>>> metadata = c.geospatial.put_raster(feature_type.external_id, feature.external_
↪id,
...     raster_property_name, "XYZ", 3857, file)
```

Delete raster data

`GeospatialAPI.delete_raster` (*feature_type_external_id: str, feature_external_id: str, raster_property_name: str*) → None

Delete raster <<https://docs.cognite.com/api/v1/#tag/Geospatial/operation/deleteRaster>>

Parameters

- **feature_type_external_id** – Feature type definition for the features to create.
- **feature_external_id** – one feature or a list of features to create
- **raster_property_name** – the raster property name

Returns None

Examples

Delete a raster in a feature raster property:

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

(continues on next page)

(continued from previous page)

```
>>> raster_property_name = ...
>>> c.geospatial.delete_raster(feature_type.external_id, feature.external_id,
↳ raster_property_name)
```

Get raster data

`GeospatialAPI.get_raster` (*feature_type_external_id: str, feature_external_id: str, raster_property_name: str, raster_format: str, raster_options: Optional[Dict[str, Any]] = None, raster_srid: Optional[int] = None, raster_scale_x: Optional[float] = None, raster_scale_y: Optional[float] = None, allow_crs_transformation: bool = False*) → bytes

Get raster <<https://docs.cognite.com/api/v1/#tag/Geospatial/operation/getRaster>>

Parameters

- **feature_type_external_id** – Feature type definition for the features to create.
- **feature_external_id** – one feature or a list of features to create
- **raster_property_name** – the raster property name
- **raster_format** – the raster output format
- **raster_options** – GDAL raster creation key-value options
- **raster_srid** – the SRID for the output raster
- **raster_scale_x** – the X component of the output pixel width in units of coordinate reference system
- **raster_scale_y** – the Y component of the output pixel height in units of coordinate reference system
- **allow_crs_transformation** – When the parameter is false, requests with output rasters in Coordinate Reference System different from the one defined in the feature type will result in bad request response code.

Returns the raster data

Return type bytes

Examples

Get a raster from a feature raster property:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> feature_type = ...
>>> feature = ...
>>> raster_property_name = ...
>>> raster_data = c.geospatial.get_raster(feature_type.external_id, feature.
↳ external_id,
... raster_property_name, "XYZ", {"SIGNIFICANT_DIGITS": "4"})
```

Compute

`GeospatialAPI.compute` (*output: Dict[str, cognite.client.data_classes.geospatial.GeospatialComputeFunction]*)
 → `cognite.client.data_classes.geospatial.GeospatialComputedResponse`

Compute <<https://docs.cognite.com/api/v1/#tag/Geospatial/operation/compute>>

Parameters `output` – Mapping of keys to compute functions.

Returns Mapping of keys to computed items.

Return type dict

Examples

Compute the transformation of an ewkt geometry from one SRID to another:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes.geospatial import _
↳ GeospatialGeometryTransformComputeFunction, _
↳ GeospatialGeometryValueComputeFunction
>>> c = CogniteClient()
>>> compute_function = _
↳ GeospatialGeometryTransformComputeFunction(GeospatialGeometryValueComputeFunction(
↳ "SRID=4326;POLYGON((0 0,10 0,10 10,0 10,0 0))"), srid=23031)
>>> compute_result = c.geospatial.compute(output = {"output": compute_function})
```

Data classes

```
class cognite.client.data_classes.geospatial.CoordinateReferenceSystem (srid:
                                                                    int
                                                                    =
                                                                    None,
                                                                    wkt:
                                                                    str
                                                                    =
                                                                    None,
                                                                    proj_string:
                                                                    str
                                                                    =
                                                                    None,
                                                                    cog-
                                                                    nite_client:
                                                                    Cog-
                                                                    nite-
                                                                    Client
                                                                    =
                                                                    None)
```

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

A representation of a feature in the geospatial api.

```
class cognite.client.data_classes.geospatial.CoordinateReferenceSystemList (resources:
                                                                    Collec-
                                                                    tion[Any],
                                                                    cog-
                                                                    nite_client:
                                                                    Cognite-
                                                                    Client
                                                                    =
                                                                    None)
```

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

```
class cognite.client.data_classes.geospatial.Feature (external_id: str = None, cog-
                                                                    nite_client: CogniteClient =
                                                                    None, **properties)
```

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

A representation of a feature in the geospatial api.

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]

```
class cognite.client.data_classes.geospatial.FeatureAggregate (cognite_client:
                                                                    CogniteClient =
                                                                    None)
```

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

A result of aggregating features in geospatial api.

```
class cognite.client.data_classes.geospatial.FeatureAggregateList (resources:
                                                                    Collec-
                                                                    tion[Any],
                                                                    cog-
                                                                    nite_client:
                                                                    Cognite-
                                                                    Client
                                                                    =
                                                                    None)
```

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

```
class cognite.client.data_classes.geospatial.FeatureList (resources: Collec-
                                                                    tion[Any], cog-
                                                                    nite_client: Cognite-
                                                                    Client = None)
```

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

```
static from_geopandas (feature_type: FeatureType, geodataframe: geopandas.GeoDataFrame,
                                                                    external_id_column: str = 'externalId', property_column_mapping:
                                                                    Dict[str, str] = None, data_set_id_column: str = 'dataSetId') → Fea-
                                                                    tureList
```

Convert a GeoDataFrame instance into a FeatureList.

Parameters

- **feature_type** (*FeatureType*) – The feature type the features will conform to
- **geodataframe** (*GeoDataFrame*) – the geodataframe instance to convert into features

- **external_id_column** – the geodataframe column to use for the feature external id
- **data_set_id_column** – the geodataframe column to use for the feature dataSet id
- **property_column_mapping** – provides a mapping from featuretype property names to geodataframe columns

Returns The list of features converted from the geodataframe rows.

Return type *FeatureList*

Examples

Create features from a geopandas dataframe:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> my_feature_type = ... # some feature type with 'position' and 'temperature
↳ 'properties
>>> my_geodataframe = ... # some geodataframe with 'center_xy', 'temp' and
↳ 'id' columns
>>> feature_list = FeatureList.from_geopandas(feature_type=my_feature_type,
↳ geodataframe=my_geodataframe,
>>>     external_id_column="id", data_set_id_column="dataSetId",
>>>     property_column_mapping={'position': 'center_xy', 'temperature': 'temp
↳'})
>>> created_features = c.geospatial.create_features(my_feature_type.external_
↳id, feature_list)
```

to_geopandas (*geometry: str, camel_case: bool = False*) → *geopandas.GeoDataFrame*

Convert the instance into a GeoPandas GeoDataFrame.

Parameters

- **geometry** (*str*) – The name of the feature type geometry property to use in the GeoDataFrame
- **camel_case** (*bool*) – Convert column names to camel case (e.g. *externalId* instead of *external_id*)

Returns The GeoPandas GeoDataFrame.

Return type *geopandas.GeoDataFrame*

Examples

Convert a FeatureList into a GeoPandas GeoDataFrame:

```
>>> from cognite.client.data_classes.geospatial import PropertyAndSearchSpec
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> features = c.geospatial.search_features(...)
>>> gdf = features.to_geopandas(
...     geometry="position",
...     camel_case=False
... )
>>> gdf.head()
```

```
class cognite.client.data_classes.geospatial.FeatureType (external_id: str = None,  
data_set_id: int = None,  
created_time: int = None,  
last_updated_time: int  
= None, properties:  
Dict[str, Any] = None,  
search_spec: Dict[str,  
Any] = None, cog-  
nite_client: Cognite-  
Client = None)
```

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

A representation of a feature type in the geospatial api.

```
class cognite.client.data_classes.geospatial.FeatureTypeList (resources: Col-  
lection[Any],  
cognite_client:  
CogniteClient =  
None)
```

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

```
class cognite.client.data_classes.geospatial.FeatureTypePatch (external_id:  
'Optional[str]'  
= None, prop-  
erty_patches:  
'Op-  
tional[Patches]'  
= None,  
search_spec_patches:  
'Op-  
tional[Patches]'  
= None)
```

Bases: `object`

```
class cognite.client.data_classes.geospatial.FeatureTypeUpdate (external_id: str  
= None, add:  
PropertyAnd-  
SearchSpec =  
None, remove:  
PropertyAnd-  
SearchSpec  
= None, cog-  
nite_client:  
CogniteClient =  
None)
```

Bases: `object`

A representation of a feature type update in the geospatial api.

```
class cognite.client.data_classes.geospatial.GeospatialComputeFunction
```

Bases: `abc.ABC`

A geospatial compute function

```
to_json_payload() → dict  
Convert function to json for request payload
```

```
class cognite.client.data_classes.geospatial.GeospatialComputedItem (resource:
    Dict[str,
    Any],
    cog-
    nite_client:
    Cognite-
    Client =
    None)
```

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

A representation of an item computed from geospatial.

```
class cognite.client.data_classes.geospatial.GeospatialComputedItemList (resources:
    Col-
    lec-
    tion[Any],
    cog-
    nite_client:
    Cog-
    nite-
    Client
    =
    None)
```

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

A list of items computed from geospatial.

```
class cognite.client.data_classes.geospatial.GeospatialComputedResponse (computed_item_list:
    Geospa-
    tial-
    Com-
    pute-
    dItem-
    List,
    cog-
    nite_client:
    Cog-
    nite-
    Client
    =
    None)
```

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

The geospatial compute response.

```
class cognite.client.data_classes.geospatial.GeospatialGeometryComputeFunction
    Bases: cognite.client.data_classes.geospatial.GeospatialComputeFunction, abc.
    ABC
```

A geospatial geometry compute function

```
class cognite.client.data_classes.geospatial.GeospatialGeometryTransformComputeFunction (geo
    cog-
    nite
    sria
    int)
```

Bases: *cognite.client.data_classes.geospatial.GeospatialComputeFunction*

A stTransform geospatial compute function

`to_json_payload()` → dict
Convert function to json for request payload

class `cognite.client.data_classes.geospatial.GeospatialGeometryValueComputeFunction` (*ewkt: str*)

Bases: `cognite.client.data_classes.geospatial.GeospatialGeometryComputeFunction`

A geospatial geometry value compute function. Accepts a well-known text of the geometry prefixed with a spatial reference identifier, see <https://docs.geotools.org/stable/javadocs/org/opengis/referencing/doc-files/WKT.html>

`to_json_payload()` → dict
Convert function to json for request payload

class `cognite.client.data_classes.geospatial.OrderSpec` (*property: str, direction: str*)

Bases: object

An order specification with respect to an property.

class `cognite.client.data_classes.geospatial.Patches` (*add: Optional[Dict[str, Any]] = None, remove: Optional[List[str]] = None*)

Bases: object

class `cognite.client.data_classes.geospatial.PropertyAndSearchSpec` (*properties: Union[Dict[str, Any], List[str], None] = None, search_spec: Union[Dict[str, Any], List[str], None] = None*)

Bases: object

A representation of a feature type property and search spec.

class `cognite.client.data_classes.geospatial.RasterMetadata` (***properties*)

Bases: object

Raster metadata

`cognite.client.data_classes.geospatial.nan_to_none` (*column_value: Any*) → Any
Convert NaN value to None.

2.4.16 3D

Models

Retrieve a model by ID

`ThreeDModelsAPI.retrieve` (*id: int*) → `Optional[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: Optional[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, Sequence[str]]*) → `cognite.client.data_classes.three_d.ThreeDModelList`

Create new 3d models.

Parameters `name` (`Union[str, Sequence[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, Sequence[Union[cognite.client.data_classes.three_d.ThreeDModel, cognite.client.data_classes.three_d.ThreeDModelUpdate]]]`) → `Union[cognite.client.data_classes.three_d.ThreeDModel, cognite.client.data_classes.three_d.ThreeDModelList]`

Update 3d models.

Parameters `item` (`Union[ThreeDModel, ThreeDModelUpdate, Sequence[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, Sequence[int]]`) → `None`
Delete 3d models.

Parameters `id` (`Union[int, Sequence[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`) → `Optional[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 `Optional[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, Sequence[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, Sequence[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, Sequence[Union[cognite.client.data_classes.three_d.ThreeDModelRevision, cognite.client.data_classes.three_d.ThreeDModelRevisionUpdate]]]*) → `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, Sequence[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)
>>> 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, Sequence[int]]) → None`
Delete 3d model revisions.

Parameters

- **model_id** (*int*) – Delete the revision under the model with this id.
- **id** (*Union[int, Sequence[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: Optional[int] = None, depth: Optional[int] = None, sort_by_node_id: bool = False, partitions: Optional[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*) – Maximum number of nodes to return. Defaults to 25. Set to -1, float(“inf”) or None to return all items.
- **sort_by_node_id** (*bool*) – Returns the nodes in *nodeId* order.
- **partitions** (*int*) – The result is retrieved in this many parts in parallel. Requires *sort_by_node_id* to be set to *true*.

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)
```

Filter nodes

`ThreeDRevisionsAPI.filter_nodes` (*model_id: int, revision_id: int, properties: Optional[Dict[str, Dict[str, Sequence[str]]]] = None, limit: int = 25, partitions: Optional[int] = None*) → `cognite.client.data_classes.three_d.ThreeDNodeList`

List nodes in a revision, filtered by node property values.

Parameters

- **model_id** (*int*) – Id of the model.
- **revision_id** (*int*) – Id of the revision.
- **properties** (*Dict[str, Dict[str, Sequence[str]]]*) – Properties for filtering. The object contains one or more category. Each category references one or more properties. Each property is associated with a list of values. For a node to satisfy the filter, it must, for each category/property in the filter, contain the category+property combination with a value that is contained within the corresponding list in the filter.
- **limit** (*int*) – Maximum number of nodes to return. Defaults to 25. Set to -1, float("inf") or None to return all items.
- **partitions** (*int*) – The result is retrieved in this many parts in parallel. Requires *sort_by_node_id* to be set to *true*.

Returns The list of 3d nodes.

Return type *ThreeDNodeList*

Example

Filter nodes from the hierarchy in the 3d model that have one of the values “AB76”, “AB77” or “AB78” for property PDMS/Area AND that also have one of the values “PIPE”, “BEND” or “PIPESUP” for the property PDMS/Type.

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.three_d.revisions.filter_nodes(model_id=1, revision_id=1, properties=
↳ { "PDMS": { "Area": ["AB76", "AB77", "AB78"], "Type": ["PIPE", "BEND", "PIPESUP
↳ "] } }, limit=10)
```

List ancestor nodes

`ThreeDRevisionsAPI.list_ancestor_nodes` (*model_id: int, revision_id: int, node_id: Optional[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

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*) – Maximum 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, Sequence[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, Sequence[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: Optional[int] = None, asset_id: Optional[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, Sequence[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, Sequence[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: Optional[List[float]] = None, min: Optional[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: Optional[List[float]] = None, position: Optional[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: CogniteClient = 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: Collection[Any], cognite_client: CogniteClient = 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: CogniteClient = 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:      Collec-
                                                    tion[Any],      cog-
                                                    nite_client:    Cog-
                                                    niteClient = 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, rota-
tion: List[float]
= None, camera:
Union[Dict[str,
Any], Revision-
CameraProper-
ties] = None,
status: str =
None, metadata:
Dict[str, str]
= None, thumb-
nail_threed_file_id:
int = None,
thumbnail_url:
str = None, as-
set_mapping_count:
int = None, cre-
ated_time: int
= None, cog-
nite_client:
CogniteClient =
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:
    Collection[Any],
    cognite_client:
    CogniteClient =
    None)
```

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

```
class cognite.client.data_classes.three_d.ThreeDModelRevisionUpdate (id: Optional[int]
    = None,
    external_id:
    Optional[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: Optional[int] =
    None, external_id:
    Optional[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],
    BoundingBox3D] = None, cognite_client:
    CogniteClient =
    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:          Col-
                                                         lection[Any],          cog-
                                                         nite_client:          Cognite-
                                                         Client = None)
Bases: cognite.client.data_classes._base.CogniteResourceList
```

2.4.17 Contextualization

These APIs will return as soon as possible, deferring a blocking wait until the last moment. Nevertheless, they can block for a long time awaiting results.

Fit Entity Matching Model

```
EntityMatchingAPI.fit (sources:          Sequence[Union[Dict[KT,          VT],          cog-
nrite.client.data_classes._base.CogniteResource]],          tar-
gets:          Sequence[Union[Dict[KT,          VT],          cog-
nrite.client.data_classes._base.CogniteResource]],          true_matches:          Op-
tional[Sequence[Union[Dict[KT,          VT],          Tuple[Union[int,          str],          Union[int,
str]]]]] = None, match_fields:          Union[Dict[KT,          VT],          Sequence[Tuple[str,
str]],          None] = None, feature_type:          Optional[str] = None, classifier:          Op-
tional[str] = None, ignore_missing_fields:          bool = False, name:          Optional[str]
= None, description:          Optional[str] = None, external_id:          Optional[str] =
None) → cognite.client.data_classes.contextualization.EntityMatchingModel
```

Fit entity matching model.

Note: All users on this CDF subscription with assets read-all and entitymatching read-all and write-all capabilities in the project, are able to access the data sent to this endpoint.

Parameters

- **sources** – entities to match from, should have an ‘id’ field. Tolerant to passing more than is needed or used (e.g. json dump of time series list). Metadata fields are automatically flattened to “metadata.key” entries, such that they can be used in match_fields.
- **targets** – entities to match to, should have an ‘id’ field. Tolerant to passing more than is needed or used.
- **true_matches** – Known valid matches given as a list of dicts with keys ‘sourceId’, ‘sourceExternalId’, ‘targetId’, ‘targetExternalId’. If omitted, uses an unsupervised model.

A tuple can be used instead of the dictionary for convenience, interpreted as id/externalId based on type.

- **match_fields** – List of (from,to) keys to use in matching. Default in the API is [(‘name’,‘name’)]. Also accepts {“source”: .., “target”: ..}.
- **feature_type** (*str*) – feature type that defines the combination of features used, see API docs for details.
- **classifier** (*str*) – classifier used in training.
- **ignore_missing_fields** (*bool*) – whether missing data in match_fields should return error or be filled in with an empty string.
- **name** (*str*) – Optional user-defined name of model.
- **description** (*str*) – Optional user-defined description of model.
- **external_id** (*str*) – Optional external id. Must be unique within the project.

Returns Resulting queued model.

Return type *EntityMatchingModel*

Re-fit Entity Matching Model

`EntityMatchingAPI.refit` (*true_matches*: *Sequence[Union[Dict[KT, VT], Tuple[Union[int, str], Union[int, str]]]]*, *id*: *Optional[int]* = *None*, *external_id*: *Optional[str]* = *None*) → `cognite.client.data_classes.contextualization.EntityMatchingModel`

Re-fits an entity matching model, using the combination of the old and new true matches.

Note: All users on this CDF subscription with assets read-all and entitymatching read-all and write-all capabilities in the project, are able to access the data sent to this endpoint.

Parameters

- **true_matches** (*Sequence[Union[Dict, Tuple[Union[int, str], Union[int, str]]]]*) – Updated known valid matches given as a list of dicts with keys ‘fromId’, ‘fromExternalId’, ‘toId’, ‘toExternalId’). A tuple can be used instead of the dictionary for convenience, interpreted as id/externalId based on type.
- **id** – ids of the model to use.
- **external_id** – external ids of the model to use.

Returns new model refitted to true_matches.

Return type *EntityMatchingModel*

Retrieve Entity Matching Models

`EntityMatchingAPI.retrieve` (*id*: *Optional[int]* = *None*, *external_id*: *Optional[str]* = *None*) → `Optional[cognite.client.data_classes.contextualization.EntityMatchingModel]`

Retrieve model

Parameters

- **id** (*int*) – id of the model to retrieve.

- **external_id** (*str*) – external id of the model to retrieve.

Returns Model requested.

Return type *EntityMatchingModel*

`EntityMatchingAPI.retrieve_multiple` (*ids*: *Optional[Sequence[int]] = None*, *external_ids*: *Optional[Sequence[str]] = None*) → `cognite.client.data_classes.contextualization.EntityMatchingModelList`

Retrieve models

Parameters

- **ids** (*Sequence[int]*) – ids of the model to retrieve.
- **external_ids** (*Sequence[str]*) – external ids of the model to retrieve.

Returns Models requested.

Return type *EntityMatchingModelList*

`EntityMatchingAPI.list` (*name*: *Optional[str] = None*, *description*: *Optional[str] = None*, *original_id*: *Optional[int] = None*, *feature_type*: *Optional[str] = None*, *classifier*: *Optional[str] = None*, *limit*: *int = 100*) → `cognite.client.data_classes.contextualization.EntityMatchingModelList`

List models

Parameters

- **name** (*str*) – Optional user-defined name of model.
- **description** (*str*) – Optional user-defined description of model.
- **feature_type** (*str*) – feature type that defines the combination of features used.
- **classifier** (*str*) – classifier used in training.
- **original_id** (*int*) – id of the original model for models that were created with refit.
- **limit** (*int, optional*) – Maximum number of items to return. Defaults to 100. Set to -1, float("inf") or None to return all items.

Returns List of models.

Return type *EntityMatchingModelList*

Delete Entity Matching Models

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

Delete models

Parameters

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

Update Entity Matching Models

`EntityMatchingAPI.update` (*item*: `Union[cognite.client.data_classes.contextualization.EntityMatchingModel, cognite.client.data_classes.contextualization.EntityMatchingModelUpdate, Sequence[Union[cognite.client.data_classes.contextualization.EntityMatchingModel, cognite.client.data_classes.contextualization.EntityMatchingModelUpdate]]]`)
 → `Union[cognite.client.data_classes.contextualization.EntityMatchingModelList, cognite.client.data_classes.contextualization.EntityMatchingModel]`

Update model

Parameters *item* (`Union[EntityMatchingModel, EntityMatchingModelUpdate, Sequence[Union[EntityMatchingModel, EntityMatchingModelUpdate]]]`)
 – Model(s) to update

Predict Using an Entity Matching Model

`EntityMatchingAPI.predict` (*sources*: `Optional[Sequence[Dict[KT, VT]]] = None`, *targets*: `Optional[Sequence[Dict[KT, VT]]] = None`, *num_matches*: `int = 1`, *score_threshold*: `Optional[float] = None`, *id*: `Optional[int] = None`, *external_id*: `Optional[str] = None`) → `cognite.client.data_classes.contextualization.ContextualizationJob`

Predict entity matching.

Warning: Blocks and waits for the model to be ready if it has been recently created.

Note: All users on this CDF subscription with assets read-all and entitymatching read-all and write-all capabilities in the project, are able to access the data sent to this endpoint.

Parameters

- **sources** (`Optional[Sequence[Dict]]`) – entities to match from, does not need an ‘id’ field. Tolerant to passing more than is needed or used (e.g. json dump of time series list). If omitted, will use data from fit.
- **targets** (`Optional[Sequence[Dict]]`) – entities to match to, does not need an ‘id’ field. Tolerant to passing more than is needed or used. If omitted, will use data from fit.
- **num_matches** (`int`) – number of matches to return for each item.
- **score_threshold** (`float`) – only return matches with a score above this threshold
- **ignore_missing_fields** (`bool`) – whether missing data in match_fields should be filled in with an empty string.
- **id** – ids of the model to use.
- **external_id** – external ids of the model to use.

Returns Object which can be used to wait for and retrieve results.

Return type `ContextualizationJob`

Detect entities in Engineering Diagrams

DiagramsAPI.**detect** (*entities: Sequence[Union[dict, cognite.client.data_classes._base.CogniteResource]], search_field: str = 'name', partial_match: bool = False, min_tokens: int = 2, file_ids: Union[int, Sequence[int], None] = None, file_external_ids: Union[str, Sequence[str], None] = None, file_references: Union[List[cognite.client.data_classes.contextualization.FileReference], cognite.client.data_classes.contextualization.FileReference, None] = None, *, multiple_jobs: bool = False) → Union[cognite.client.data_classes.contextualization.DiagramDetectResults, Tuple[Optional[cognite.client.data_classes.contextualization.DetectJobBundle], List[Dict[str, Any]]]]*

Detect entities in a PNID. The results are not written to CDF.

Note: All users on this CDF subscription with assets read-all and files read-all capabilities in the project, are able to access the data sent to this endpoint.

Parameters

- **entities** (*Sequence[Union[dict, CogniteResource]]*) – List of entities to detect
- **search_field** (*str*) – If entities is a list of dictionaries, this is the key to the values to detect in the PnId
- **partial_match** (*bool*) – Allow for a partial match (e.g. missing prefix).
- **min_tokens** (*int*) – Minimal number of tokens a match must be based on
- **file_ids** (*Sequence[int]*) – ID of the files, should already be uploaded in the same tenant.
- **file_external_ids** (*Sequence[str]*) – File external ids.
- **file_references** (*Sequence[FileReference]*) – File references (id or external id) with page ranges.

Keyword Arguments **multiple_jobs** (*bool*) – Enables you to publish multiple jobs. If True the method will return a tuple of DetectJobBundle and list of potentially unposted files. If False it will return a single DiagramDetectResults. Defaults to False.

Returns Resulting queued job or a bundle of jobs and a list of unposted files. Note that the .result property of the job or job bundle will block waiting for results.

Return type Union[*DiagramDetectResults*, Tuple[*DetectJobBundle*, List[Dict[str, Any]]]]

Examples

```
>>> from cognite.client import CogniteClient
>>> client = CogniteClient()
>>> retrieved_model = client.diagrams.detect(
    entities=[{"userDefinedField": "21PT1017", "ignoredField": "AA11"}],
    search_field="userDefinedField",
    partial_match=True,
    min_tokens=2,
    file_ids=[101],
```

(continues on next page)

(continued from previous page)

```

file_external_ids=["Test1"],
)

```

Convert to an interactive SVG where the provided annotations are highlighted

DiagramsAPI.**convert** (*detect_job*: *cognite.client.data_classes.contextualization.DiagramDetectResults*)
→ *cognite.client.data_classes.contextualization.DiagramConvertResults*

Convert a P&ID to interactive SVGs where the provided annotations are highlighted.

Parameters **detect_job** (*DiagramConvertResults*) – detect job

Returns Resulting queued job. Note that .result property of this job will block waiting for results.

Return type *DiagramConvertResults*

Examples

```

>>> from cognite.client import CogniteClient
>>> client = CogniteClient()
>>> detect_job = client.diagrams.detect(...)
>>> client.diagrams.convert(detect_job=detect_job)

```

Vision

The Vision API enable extraction of information from imagery data based on their visual content. For example, you can can extract features such as text, asset tags or industrial objects from images using this service.

Quickstart

Start an asynchronous job to extract information from image files stored in CDF:

```

from cognite.client import CogniteClient
from cognite.client.data_classes.contextualization import VisionFeature

c = CogniteClient()
extract_job = c.vision.extract(
    features=[VisionFeature.ASSET_TAG_DETECTION, VisionFeature.PEOPLE_DETECTION],
    file_ids=[1, 2],
)

```

The returned job object, `extract_job`, can be used to retrieve the status of the job and the prediction results once the job is completed. Wait for job completion and get the parsed results:

```

extract_job.wait_for_completion()
for item in extract_job.items:
    predictions = item.predictions
    # do something with the predictions

```

Save the prediction results in CDF as Annotations:

```

extract_job.save_predictions()

```

Note: Prediction results are stored in CDF as [Annotations](#) using the `images.*` annotation types. In particular, text detections are stored as `images.TextRegion`, asset tag detections are stored as `images.AssetLink`, while other detections are stored as `images.ObjectDetection`.

Tweaking the parameters of a feature extractor:

```
from cognite.client.data_classes.contextualization import FeatureParameters, \
↳TextDetectionParameters

extract_job = c.vision.extract(
    features=VisionFeature.TEXT_DETECTION,
    file_ids=[1, 2],
    parameters=FeatureParameters(text_detection_
↳parameters=TextDetectionParameters(threshold=0.9))
)
```

Extract

`VisionAPI.extract` (*features*: `Union[cognite.client.data_classes.contextualization.VisionFeature, List[cognite.client.data_classes.contextualization.VisionFeature]]`, *file_ids*: `Optional[List[int]] = None`, *file_external_ids*: `Optional[List[str]] = None`, *parameters*: `Optional[cognite.client.data_classes.contextualization.FeatureParameters] = None`) → `cognite.client.data_classes.contextualization.VisionExtractJob`

Start an asynchronous job to extract features from image files.

Parameters

- **features** (`Union[VisionFeature, List[VisionFeature]]`) – The feature(s) to extract from the provided image files.
- **file_ids** (`List[int]`) – IDs of the image files to analyze. The images must already be uploaded in the same CDF project.
- **file_external_ids** (`List[str]`) – The external file ids of the image files to analyze.

Returns Resulting queued job, which can be used to retrieve the status of the job or the prediction results if the job is finished. Note that `.result` property of this job will wait for the job to finish and returns the results.

Return type `VisionExtractJob`

Examples

Start a job, wait for completion and then get the parsed results:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes.contextualization import VisionFeature
>>> c = CogniteClient()
>>> extract_job = c.vision.extract(features=VisionFeature.ASSET_TAG_DETECTION, \
↳file_ids=[1])
>>> extract_job.wait_for_completion()
>>> for item in extract_job.items:
...     predictions = item.predictions
```

(continues on next page)

(continued from previous page)

```

...     # do something with the predictions
>>> # Save predictions in CDF using Annotations API:
>>> extract_job.save_predictions()

```

Get vision extract job

`VisionAPI.get_extract_job(job_id: int) → cognite.client.data_classes.contextualization.VisionExtractJob`
Retrieve an existing extract job by ID.

Parameters `job_id(int)` – ID of an existing feature extraction job.

Returns Vision extract job, which can be used to retrieve the status of the job or the prediction results if the job is finished. Note that `.result` property of this job will wait for the job to finish and returns the results.

Return type `VisionExtractJob`

Examples

Retrieve a vision extract job by ID:

```

>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> extract_job = c.vision.get_extract_job(job_id=1)
>>> extract_job.wait_for_completion()
>>> for item in extract_job.items:
...     predictions = item.predictions
...     # do something with the predictions

```

Contextualization Data Classes

```

class cognite.client.data_classes.contextualization.AssetTagDetectionParameters (*args:
                                                                              'Any',
                                                                              **kwargs:
                                                                              'Any')
    →
    'CogniteResource'

```

Bases: `cognite.client.data_classes.annotation_types.primitives.VisionResource`, `cognite.client.data_classes.contextualization.ThresholdParameter`

asset_subtree_ids = None

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]`

partial_match = None

threshold = None

to_pandas (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Parameters **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.contextualization.ContextualizationJob (job_id:
    int
    =
    None,
    model_id:
    int
    =
    None,
    sta-
    tus:
    str
    =
    None,
    er-
    ror_message:
    str
    =
    None,
    cre-
    ated_time:
    int
    =
    None,
    start_time:
    int
    =
    None,
    sta-
    tus_time:
    int
    =
    None,
    sta-
    tus_path:
    str
    =
    None,
    job_token:
    str
    =
    None,
    cog-
    nite_client:
    Cog-
    nite-
    Client
    =
    None)

```

Bases: `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]

result

Waits for the job to finish and returns the results.

to_pandas (*expand: Sequence[str] = ('metadata',), ignore: List[str] = None, camel_case: bool = False*) → pandas.DataFrame

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

update_status () → str

Updates the model status and returns it

wait_for_completion (*timeout: Optional[int] = None, interval: int = 1*) → None

Waits for job completion. This is generally not needed to call directly, as *.result* will do so automatically.

Parameters

- **timeout** (*int*) – Time out after this many seconds. (None means wait indefinitely)
- **interval** (*int*) – Poll status every this many seconds.

Raises ModelFailedException – The model fit failed.

```
class cognite.client.data_classes.contextualization.ContextualizationJobList (resources: Col-  
lec-  
tion[Any],  
cog-  
nite_client:  
Cog-  
nite-  
Client  
=  
None)
```

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

append (*item*)

S.append(value) – append value to the end of the sequence

clear () → None – remove all items from S

copy ()

count (*value*) → integer – return number of occurrences of value

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]]

extend (*other: Collection[Any]*) → None

S.extend(iterable) – extend sequence by appending elements from the iterable

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

index (*value* [, *start* [, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

insert (*i, item*)

S.insert(index, value) – insert value before index

pop ([*index*]) → item – remove and return item at index (default last).
Raise IndexError if list is empty or index is out of range.

remove (*item*)

S.remove(value) – remove first occurrence of value. Raise ValueError if the value is not present.

reverse ()

S.reverse() – reverse *IN PLACE*

sort (**args, **kwds*)

to_pandas (*camel_case: bool = False*) → pandas.DataFrame
Convert the instance into a pandas DataFrame.

Returns The dataframe.

Return type pandas.DataFrame

class cognite.client.data_classes.contextualization.**ContextualizationJobType**

Bases: enum.Enum

An enumeration.

DIAGRAMS = 'diagrams'

ENTITY_MATCHING = 'entity_matching'

VISION = 'vision'

class cognite.client.data_classes.contextualization.**DetectJobBundle** (*job_ids: List[int], cognite_client: CogniteClient = None*)

Bases: object

fetch_results () → List[Dict[str, Any]]

result

Waits for the job to finish and returns the results.

wait_for_completion (*timeout: Optional[int] = None*) → None

Waits for all jobs to complete, generally not needed to call as it is called by result.

Parameters

- **timeout** (*int*) – Time out after this many seconds. (None means wait indefinitely)
- **interval** (*int*) – Poll status every this many seconds.

```
class cognite.client.data_classes.contextualization.DiagramConvertItem (file_id:  
    int  
    =  
    None,  
    file_external_id:  
    str  
    =  
    None,  
    re-  
    sults:  
    list  
    =  
    None,  
    cog-  
    nite_client:  
    Cog-  
    nite-  
    Client  
    =  
    None)
```

Bases: *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]

pages

to_pandas (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Parameters **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.contextualization.DiagramConvertPage (page:
                                                                    int
                                                                    =
                                                                    None,
                                                                    png_url:
                                                                    str
                                                                    =
                                                                    None,
                                                                    svg_url:
                                                                    str
                                                                    =
                                                                    None,
                                                                    cog-
                                                                    nite_client:
                                                                    Cog-
                                                                    nite-
                                                                    Client
                                                                    =
                                                                    None)

```

Bases: `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: Sequence[str] = ('metadata',), ignore: List[str] = None, camel_case: bool = False*) → pandas.DataFrame

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

```

class cognite.client.data_classes.contextualization.DiagramConvertPageList (resources:
                                                                              Col-
                                                                              lec-
                                                                              tion[Any],
                                                                              cog-
                                                                              nite_client:
                                                                              Cog-
                                                                              nite-
                                                                              Client
                                                                              =
                                                                              None)

```

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

append (*item*)

S.append(value) – append value to the end of the sequence

clear () → None – remove all items from S

copy ()

count (*value*) → integer – return number of occurrences of value

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]]

extend (*other: Collection[Any]*) → None

S.extend(iterable) – extend sequence by appending elements from the iterable

get (*id: Optional[int] = None, external_id: Optional[str] = None*) → Op-

tional[cognite.client.data_classes._base.CogniteResource]
Get an item from this list by id or external_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]

index (*value* [, *start* [, *stop*]]) → integer – return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

insert (*i, item*)

S.insert(index, value) – insert value before index

pop ([*index*]) → item – remove and return item at index (default last).

Raise IndexError if list is empty or index is out of range.

remove (*item*)

S.remove(value) – remove first occurrence of value. Raise ValueError if the value is not present.

reverse ()

S.reverse() – reverse *IN PLACE*

sort (**args, **kws*)

to_pandas (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Returns The dataframe.

Return type pandas.DataFrame

class cognite.client.data_classes.contextualization.DiagramConvertResults (***kwargs*)

Bases: *cognite.client.data_classes.contextualization.ContextualizationJob*

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]

items

returns a list of all results by file

result

Waits for the job to finish and returns the results.

to_pandas (*expand: Sequence[str] = ('metadata',), ignore: List[str] = None, camel_case: bool = False*) → pandas.DataFrame

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

update_status () → str

Updates the model status and returns it

wait_for_completion (*timeout: Optional[int] = None, interval: int = 1*) → None

Waits for job completion. This is generally not needed to call directly, as *.result* will do so automatically.

Parameters

- **timeout** (*int*) – Time out after this many seconds. (None means wait indefinitely)
- **interval** (*int*) – Poll status every this many seconds.

Raises ModelFailedException – The model fit failed.

```

class cognite.client.data_classes.contextualization.DiagramDetectItem (file_id:
                                                                    int =
                                                                    None,
                                                                    file_external_id:
                                                                    str =
                                                                    None,
                                                                    an-
                                                                    nota-
                                                                    tions:
                                                                    list =
                                                                    None,
                                                                    er-
                                                                    ror_message:
                                                                    str =
                                                                    None,
                                                                    cog-
                                                                    nite_client:
                                                                    Cog-
                                                                    nite-
                                                                    Client
                                                                    =
                                                                    None,
                                                                    page_range:
                                                                    Op-
                                                                    tional[Dict[str,
                                                                    int]]
                                                                    =
                                                                    None)

```

Bases: `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 (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Parameters **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.contextualization.DiagramDetectResults (**kwargs)

```

Bases: `cognite.client.data_classes.contextualization.ContextualizationJob`

convert () → cognite.client.data_classes.contextualization.DiagramConvertResults

Convert a P&ID to an interactive SVG where the provided annotations are highlighted

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]

errors

Returns a list of all error messages across files

items

Returns a list of all results by file

result

Waits for the job to finish and returns the results.

to_pandas (*expand: Sequence[str] = ('metadata',), ignore: List[str] = None, camel_case: bool = False*) → pandas.DataFrame

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

update_status () → str

Updates the model status and returns it

wait_for_completion (*timeout: Optional[int] = None, interval: int = 1*) → None

Waits for job completion. This is generally not needed to call directly, as *.result* will do so automatically.

Parameters

- **timeout** (*int*) – Time out after this many seconds. (None means wait indefinitely)
- **interval** (*int*) – Poll status every this many seconds.

Raises ModelFailedException – The model fit failed.

```
class cognite.client.data_classes.contextualization.EntityMatchingModel (id:
    int
    =
    None,
    sta-
    tus:
    str
    =
    None,
    er-
    ror_message:
    str
    =
    None,
    cre-
    ated_time:
    int
    =
    None,
    start_time:
    int
    =
    None,
    sta-
    tus_time:
    int
    =
    None,
    clas-
    si-
    fier:
    str
    =
    None,
    fea-
    ture_type:
    str
    =
    None,
    match_fields:
    List[str]
    =
    None,
    model_type:
    str
    =
    None,
    name:
    str
    =
    None,
    de-
    scrip-
    tion:
    str
    =
    None,
    ex-
    ter-
    nal_id:
```

Bases: `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]

predict (*sources: Optional[List[Dict[KT, VT]]] = None, targets: Optional[List[Dict[KT, VT]]] = None, num_matches: int = 1, score_threshold: Optional[float] = None*) → `cognite.client.data_classes.contextualization.ContextualizationJob`

Predict entity matching. NB. blocks and waits for the model to be ready if it has been recently created.

Parameters

- **sources** – entities to match from, does not need an ‘id’ field. Tolerant to passing more than is needed or used (e.g. json dump of time series list). If omitted, will use data from fit.
- **targets** – entities to match to, does not need an ‘id’ field. Tolerant to passing more than is needed or used. If omitted, will use data from fit.
- **num_matches** (*int*) – number of matches to return for each item.
- **score_threshold** (*float*) – only return matches with a score above this threshold
- **ignore_missing_fields** (*bool*) – whether missing data in match_fields should be filled in with an empty string.

Returns object which can be used to wait for and retrieve results.

Return type `ContextualizationJob`

refit (*true_matches: Sequence[Union[Dict[KT, VT], Tuple[Union[int, str], Union[int, str]]]]*) → `cognite.client.data_classes.contextualization.EntityMatchingModel`

Re-fits an entity matching model, using the combination of the old and new true matches.

Parameters **true_matches** – Updated known valid matches given as a list of dicts with keys ‘fromId’, ‘fromExternalId’, ‘toId’, ‘toExternalId’). A tuple can be used instead of the dictionary for convenience, interpreted as id/externalId based on type.

Returns new model refitted to true_matches.

Return type `EntityMatchingModel`

to_pandas (*expand: Sequence[str] = ('metadata',), ignore: List[str] = None, camel_case: bool = False*) → `pandas.DataFrame`

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`

update_status () → str

Updates the model status and returns it

wait_for_completion (*timeout: Optional[int] = None, interval: int = 1*) → None

Waits for model completion. This is generally not needed to call directly, as *.result* will do so automatically.

Parameters

- **timeout** – Time out after this many seconds. (None means wait indefinitely)
- **interval** – Poll status every this many seconds.

Raises `ModelFailedException` – The model fit failed.

```
class cognite.client.data_classes.contextualization.EntityMatchingModelList (resources:  
                                                                    Col-  
                                                                    lec-  
                                                                    tion[Any],  
                                                                    cog-  
                                                                    nite_client:  
                                                                    Cog-  
                                                                    nite-  
                                                                    Client  
                                                                    =  
                                                                    None)
```

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

append (*item*)

S.append(value) – append value to the end of the sequence

clear () → None – remove all items from S

copy ()

count (*value*) → integer – return number of occurrences of value

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]]

extend (*other: Collection[Any]*) → None

S.extend(iterable) – extend sequence by appending elements from the iterable

get (*id: Optional[int] = None, external_id: Optional[str] = None*) → Optional[cognite.client.data_classes._base.CogniteResource]

Get an item from this list by id or external_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`]

index (*value* [, *start* [, *stop*]]) → integer – return first index of value.

Raises `ValueError` if the value is not present.

Supporting start and stop arguments is optional, but recommended.

insert (*i, item*)

S.insert(index, value) – insert value before index

pop (*[index]*) → item – remove and return item at index (default last).

Raise IndexError if list is empty or index is out of range.

remove (*item*)

S.remove(value) – remove first occurrence of value. Raise ValueError if the value is not present.

reverse ()

S.reverse() – reverse *IN PLACE*

sort (**args, **kwds*)

to_pandas (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Returns The dataframe.

Return type pandas.DataFrame

```
class cognite.client.data_classes.contextualization.EntityMatchingModelUpdate (id:
                                                                                   Op-
                                                                                   tional[int]
                                                                                   =
                                                                                   None,
                                                                                   ex-
                                                                                   ter-
                                                                                   nal_id:
                                                                                   Op-
                                                                                   tional[str]
                                                                                   =
                                                                                   None)
```

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

Changes applied to entity matching model

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.

description

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

Dump the instance into a json serializable Python data type.

Returns A dictionary representation of the instance.

Return type Dict[str, Any]

name

```
class cognite.client.data_classes.contextualization.FeatureParameters (*args:
                                                                                   'Any',
                                                                                   **kwargs:
                                                                                   'Any')
                                                                                   →
                                                                                   'Cog-
                                                                                   niteRe-
                                                                                   source'
```

Bases: `cognite.client.data_classes.annotation_types.primitives.VisionResource`

asset_tag_detection_parameters = None

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]

industrial_object_detection_parameters = None

people_detection_parameters = None

personal_protective_equipment_detection_parameters = None

text_detection_parameters = None

to_pandas (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Parameters **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.contextualization.FileReference` (*file_id: Optional[int] = None, file_external_id: Optional[str] = None, first_page: Optional[int] = None, last_page: Optional[int] = None*)

Bases: object

to_api_item() → Dict[str, Union[str, int, Dict[str, int]]]

class `cognite.client.data_classes.contextualization.IndustrialObjectDetectionParameters` (**args, **kwargs*) → CogniteSource

Bases: `cognite.client.data_classes.annotation_types.primitives.VisionResource, cognite.client.data_classes.contextualization.ThresholdParameter`

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]

threshold = None

to_pandas (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Parameters **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.contextualization.**JobStatus**

Bases: enum.Enum

An enumeration.

COLLECTING = 'Collecting'

COMPLETED = 'Completed'

DISTRIBUTED = 'Distributed'

DISTRIBUTING = 'Distributing'

FAILED = 'Failed'

QUEUED = 'Queued'

RUNNING = 'Running'

is_finished() → bool

is_not_finished() → bool

class cognite.client.data_classes.contextualization.**PeopleDetectionParameters** (**args: 'Any', **kwargs: 'Any'*) → 'CogniteResource'

Bases: *cognite.client.data_classes.annotation_types.primitives.VisionResource*, *cognite.client.data_classes.contextualization.ThresholdParameter*

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]

threshold = None

to_pandas (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Parameters **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.contextualization.**PersonalProtectiveEquipmentDetectionPa**

Bases: *cognite.client.data_classes.annotation_types.primitives.VisionResource*, *cognite.client.data_classes.contextualization.ThresholdParameter*

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]

threshold = None

to_pandas (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Parameters **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.contextualization.**TextDetectionParameters** (**args: 'Any', **kwargs: 'Any'*) → 'CogniteResource'

Bases: *cognite.client.data_classes.annotation_types.primitives.VisionResource*, *cognite.client.data_classes.contextualization.ThresholdParameter*

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]

threshold = None

to_pandas (*camel_case: bool = False*) → pandas.DataFrame
Convert the instance into a pandas DataFrame.

Parameters **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.contextualization.ThresholdParameter (threshold:
    'Optional[float]'
    =
    None)
```

Bases: object

threshold = None

```
class cognite.client.data_classes.contextualization.VisionExtractItem (file_id:
    int =
    None,
    pre-
    dic-
    tions:
    Dict[str,
    Any]
    =
    None,
    file_external_id:
    str =
    None,
    er-
    ror_message:
    str =
    None,
    cog-
    nite_client:
    Cog-
    nite-
    Client
    =
    None)
```

Bases: *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: Sequence[str] = ('metadata',), ignore: List[str] = None, camel_case: bool = False*) → pandas.DataFrame
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

class `cognite.client.data_classes.contextualization.VisionExtractJob` (*args,
**kwargs)

Bases: `cognite.client.data_classes.contextualization.VisionJob`

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]

errors

Returns a list of all error messages across files

items

Returns a list of all predictions by file

result

Waits for the job to finish and returns the results.

save_predictions (*creating_user: Optional[str] = None, creating_app: Optional[str] = None, creating_app_version: Optional[str] = None*)
→ Union[cognite.client.data_classes.annotations.Annotation, cognite.client.data_classes.annotations.AnnotationList]

Saves all predictions made by the feature extractors in CDF using the Annotations API. See <https://docs.cognite.com/api/v1/#tag/Annotations/operation/annotationsSuggest>

Parameters

- **creating_app** (*str, optional*) – The name of the app from which this annotation was created. Defaults to ‘cognite-sdk-python’.
- **creating_app_version** (*str, optional*) – The version of the app that created this annotation. Must be a valid semantic versioning (SemVer) string. Defaults to client version.
- **creating_user** – (*str, optional*): A username, or email, or name. This is not checked nor enforced. If the value is None, it means the annotation was created by a service.

Returns (suggested) annotation(s) stored in CDF.

Return type Union[*Annotation, AnnotationList*]

to_pandas (*expand: Sequence[str] = ('metadata',), ignore: List[str] = None, camel_case: bool = False*) → pandas.DataFrame

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`

update_status () → `str`

Updates the model status and returns it

wait_for_completion (*timeout: Optional[int] = None, interval: int = 1*) → `None`

Waits for job completion. This is generally not needed to call directly, as *.result* will do so automatically.

Parameters

- **timeout** (*int*) – Time out after this many seconds. (None means wait indefinitely)
- **interval** (*int*) – Poll status every this many seconds.

Raises `ModelFailedException` – The model fit failed.

```
class cognite.client.data_classes.contextualization.VisionExtractPredictions (*args:
                                                                    'Any',
                                                                    **kwargs:
                                                                    'Any')
    →
    'Cog-
    niteRe-
    source'
```

Bases: `cognite.client.data_classes.annotation_types.primitives.VisionResource`

asset_tag_predictions = `None`

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]`

industrial_object_predictions = `None`

people_predictions = `None`

personal_protective_equipment_predictions = `None`

text_predictions = `None`

to_pandas (*camel_case: bool = False*) → `pandas.DataFrame`

Convert the instance into a pandas DataFrame.

Parameters **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.contextualization.VisionFeature
```

Bases: `str, enum.Enum`

An enumeration.

```
ASSET_TAG_DETECTION = 'AssetTagDetection'  
INDUSTRIAL_OBJECT_DETECTION = 'IndustrialObjectDetection'  
PEOPLE_DETECTION = 'PeopleDetection'  
PERSONAL_PROTECTIVE_EQUIPMENT_DETECTION = 'PersonalProtectiveEquipmentDetection'  
TEXT_DETECTION = 'TextDetection'  
  
beta_features = <bound method VisionFeature.beta_features of <enum 'VisionFeature'>>  
  
class cognite.client.data_classes.contextualization.VisionJob(job_id: int =  
    None, model_id:  
    int = None, sta-  
    tus: str = None,  
    error_message:  
    str = None, cre-  
    ated_time: int =  
    None, start_time:  
    int = None,  
    status_time: int  
    = None, sta-  
    tus_path: str =  
    None, job_token:  
    str = None,  
    cognite_client:  
    CogniteClient =  
    None)  
  
Bases: cognite.client.data_classes.contextualization.ContextualizationJob  
  
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]  
  
result  
    Waits for the job to finish and returns the results.  
  
to_pandas (expand: Sequence[str] = ('metadata', ), ignore: List[str] = None, camel_case: bool =  
    False) → pandas.DataFrame  
    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  
  
update_status () → str  
    Updates the model status and returns it
```

wait_for_completion (*timeout: Optional[int] = None, interval: int = 1*) → None

Waits for job completion. This is generally not needed to call directly, as *.result* will do so automatically.

Parameters

- **timeout** (*int*) – Time out after this many seconds. (None means wait indefinitely)
- **interval** (*int*) – Poll status every this many seconds.

Raises `ModelFailedException` – The model fit failed.

```
class cognite.client.data_classes.annotation_types.images.AssetLink (*args:
    'Any',
    **kwargs:
    'Any')
    → 'Cog-
    niteRe-
    source'
```

Bases: `cognite.client.data_classes.annotation_types.primitives.VisionResource`

confidence = None

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 (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Parameters **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.annotation_types.images.ObjectDetection (*args:
    'Any',
    **kwargs:
    'Any')
    →
    'Cog-
    niteRe-
    source'
```

Bases: `cognite.client.data_classes.annotation_types.primitives.VisionResource`

bounding_box = None

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]

polygon = None

polyline = None

to_pandas (*camel_case: bool = False*) → pandas.DataFrame
Convert the instance into a pandas DataFrame.

Parameters **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.annotation_types.images.TextRegion (*args:  
    'Any',  
    **kwargs:  
    'Any')  
    →  
    'Cog-  
niteRe-  
source'  
  
Bases:  
    cognite.client.data_classes.annotation_types.primitives.  
VisionResource
```

confidence = None

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 (*camel_case: bool = False*) → pandas.DataFrame
Convert the instance into a pandas DataFrame.

Parameters **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.annotation_types.primitives.BoundingBox (*args:  
    'Any',  
    **kwargs:  
    'Any')  
    →  
    'Cog-  
niteRe-  
source'  
  
Bases:  
    cognite.client.data_classes.annotation_types.primitives.  
VisionResource
```

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 (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Parameters **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.annotation_types.primitives.CdfResourceRef (*args:
                                                                    'Any',
                                                                    **kwargs:
                                                                    'Any')
    →
    'Cog-
    niteRe-
    source'
```

Bases: `cognite.client.data_classes.annotation_types.primitives.VisionResource`

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]

external_id = None

id = None

to_pandas (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Parameters **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.annotation_types.primitives.Point (*args:
                                                                    'Any',
                                                                    **kwargs:
                                                                    'Any')
    → 'Cog-
    niteRe-
    source'
```

Bases: `cognite.client.data_classes.annotation_types.primitives.VisionResource`

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 (*camel_case: bool = False*) → pandas.DataFrame

Convert the instance into a pandas DataFrame.

Parameters `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.annotation_types.primitives.PolyLine(*args:  
    'Any',  
    **kwargs:  
    'Any')  
    →  
    'Cog-  
    niteRe-  
    source'
```

Bases: `cognite.client.data_classes.annotation_types.primitives.VisionResource`

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 (*camel_case: bool = False*) → `pandas.DataFrame`

Convert the instance into a pandas DataFrame.

Parameters `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.annotation_types.primitives.Polygon(*args:  
    'Any',  
    **kwargs:  
    'Any')  
    →  
    'Cog-  
    niteRe-  
    source'
```

Bases: `cognite.client.data_classes.annotation_types.primitives.VisionResource`

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 (*camel_case: bool = False*) → `pandas.DataFrame`

Convert the instance into a pandas DataFrame.

Parameters `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.annotation_types.primitives.VisionResource`
 Bases: `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 (*camel_case: bool = False*) → pandas.DataFrame
 Convert the instance into a pandas DataFrame.

Parameters `camel_case` (*bool*) – Convert column names to camel case (e.g. `externalId` instead of `external_id`)

Returns The dataframe.

Return type pandas.DataFrame

2.4.18 Templates

Create Template groups

TemplateGroupsAPI.**create** (*template_groups: Union[cognite.client.data_classes.templates.TemplateGroup, Sequence[cognite.client.data_classes.templates.TemplateGroup]]*) → Union[cognite.client.data_classes.templates.TemplateGroup, cognite.client.data_classes.templates.TemplateGroupList]

Create one or more template groups.

Parameters `template_groups` (*Union[TemplateGroup, Sequence[TemplateGroup]]*) –

Returns Created template group(s)

Return type Union[*TemplateGroup, TemplateGroupList*]

Examples

Create a new template group:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TemplateGroup
>>> c = CogniteClient()
>>> template_group_1 = TemplateGroup("sdk-test-group", "This is a test group")
>>> template_group_2 = TemplateGroup("sdk-test-group-2", "This is another test_
↳group")
>>> c.templates.groups.create([template_group_1, template_group_2])
```

Upsert Template groups

TemplateGroupsAPI.**upsert** (*template_groups*: Union[cognite.client.data_classes.templates.TemplateGroup, Sequence[cognite.client.data_classes.templates.TemplateGroup]]) → Union[cognite.client.data_classes.templates.TemplateGroup, cognite.client.data_classes.templates.TemplateGroupList]

Upsert one or more template groups. Will overwrite existing template group(s) with the same external id(s).

Parameters **template_groups** (Union[TemplateGroup, Sequence[TemplateGroup]]) –

Returns Created template group(s)

Return type Union[TemplateGroup, TemplateGroupList]

Examples

Upsert a template group:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TemplateGroup
>>> c = CogniteClient()
>>> template_group_1 = TemplateGroup("sdk-test-group", "This is a test group")
>>> template_group_2 = TemplateGroup("sdk-test-group-2", "This is another test_
↳group")
>>> c.templates.groups.upsert([template_group_1, template_group_2])
```

Retrieve Template groups

TemplateGroupsAPI.**retrieve_multiple** (*external_ids*: Sequence[str], *ignore_unknown_ids*: bool = False) → cognite.client.data_classes.templates.TemplateGroupList

Retrieve multiple template groups by external id.

Parameters

- **external_ids** (Sequence[str]) – External IDs
- **ignore_unknown_ids** (bool) – Ignore external IDs that are not found rather than throw an exception.

Returns The requested template groups.

Return type TemplateGroupList

Examples

Get template groups by external id:

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

List Template groups

TemplateGroupsAPI.**list** (*limit: int = 25, owners: Optional[Sequence[str]] = None*) → `cognite.client.data_classes.templates.TemplateGroupList`

Lists template groups stored in the project based on a query filter given in the payload of this request. Up to 1000 template groups can be retrieved in one operation.

Parameters

- **owners** (*Sequence[str]*) – Include template groups that have any of these values in their *owner* field.
- **limit** (*int*) – Maximum number of template groups to return. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns List of requested template groups

Return type `TemplateGroupList`

Examples

List template groups:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> template_group_list = c.templates.groups.list(limit=5)
```

Delete Template groups

TemplateGroupsAPI.**delete** (*external_ids: Union[str, Sequence[str]], ignore_unknown_ids: bool = False*) → None

Delete one or more template groups.

Parameters

- **external_ids** (*Union[str, Sequence[str]]*) – External ID or list of external ids
- **ignore_unknown_ids** (*bool*) – Ignore external IDs that are not found rather than throw an exception.

Returns None

Examples

Delete template groups by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.templates.groups.delete(external_ids=["a", "b"])
```

Upsert a Template group version

TemplateGroupVersionsAPI.**upsert** (*external_id*: *str*, *version*: *cognite.client.data_classes.templates.TemplateGroupVersion*) → *cognite.client.data_classes.templates.TemplateGroupVersion*

Upsert a template group version. A Template Group update supports specifying different conflict modes, which is used when an existing schema already exists.

Patch -> It diffs the new schema with the old schema and fails if there are breaking changes. Update -> It sets the new schema as schema of a new version. Force -> It ignores breaking changes and replaces the old schema with the new schema. The default mode is "patch".

Parameters

- **external_id** (*str*) – The external id of the template group.
- **version** (*TemplateGroupVersion*) – The GraphQL schema of this version.

Returns Created template group version

Return type *TemplateGroupVersion*

Examples

create a new template group version modeling Covid-19:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TemplateGroup
>>> c = CogniteClient()
>>> template_group = TemplateGroup("sdk-test-group", "This template group models_
↳Covid-19 spread")
>>> c.templates.groups.create(template_group)
>>> schema = '''
>>>     type Demographics @template {
>>>         "The amount of people"
>>>         populationSize: Int,
>>>         "The population growth rate"
>>>         growthRate: Float,
>>>     }
>>>     type Country @template {
>>>         name: String,
>>>         demographics: Demographics,
>>>         deaths: TimeSeries,
>>>         confirmed: TimeSeries,
>>>     }'''
>>> template_group_version = TemplateGroupVersion(schema)
>>> c.templates.versions.upsert(template_group.external_id, template_group_
↳version)
```

List Temple Group versions

TemplateGroupVersionsAPI.**list** (*external_id*: *str*, *limit*: *int* = 25, *min_version*: *Optional[int]* = None, *max_version*: *Optional[int]* = None) → *cognite.client.data_classes.templates.TemplateGroupVersionList*

Lists versions of a specified template group. Up to 1000 template group version can be retrieved in one operation.

Parameters

- **external_id** (*str*) – The external id of the template group.
- **limit** (*int*) – Maximum number of template group versions to return. Defaults to 25. Set to -1, float("inf") or None to return all items.
- **min_version** – (Optional[int]): Exclude versions with a version number smaller than this.
- **max_version** – (Optional[int]): Exclude versions with a version number larger than this.

Returns List of requested template group versions

Return type *TemplateGroupVersionList*

Examples

List template group versions:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> template_group_list = c.templates.versions.list("template-group-ext-id",
↳ limit=5)
```

Delete a Temple Group version

`TemplateGroupVersionsAPI.delete` (*external_id: str, version: int*) → None

Delete a template group version.

Parameters

- **external_id** (*str*) – External ID of the template group.
- **version** (*int*) – The version of the template group to delete.

Returns None

Examples

Delete template groups by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.templates.versions.delete("sdk-test-group", 1)
```

Run a GraphQL query

`TemplatesAPI.graphql_query` (*external_id: str, version: int, query: str*) → `cognite.client.data_classes.templates.GraphQLResponse`

Run a GraphQL Query. To learn more, see <https://graphql.org/learn/>

Parameters

- **external_id** (*str*) – The external id of the template group.
- **version** (*int*) – The version of the template group to run the query on.
- **query** (*str*) – The GraphQL query to run.

Returns the result of the query.

Return type GraphQLResponse

Examples

Run a GraphQL query:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> query = '''
>>> {
>>>   countryQuery {
>>>     name,
>>>     demographics {
>>>       populationSize,
>>>       growthRate
>>>     },
>>>     deaths {
>>>       datapoints(limit: 100) {
>>>         timestamp,
>>>         value
>>>       }
>>>     }
>>>   }
>>> }
>>> '''
>>> result = c.templates.graphql_query("template-group-ext-id", 1, query)
```

Create Template instances

TemplateInstancesAPI.**create** (*external_id*: str, *version*: int, *instances*: Union[cognite.client.data_classes.templates.TemplateInstance, Sequence[cognite.client.data_classes.templates.TemplateInstance]])
→ Union[cognite.client.data_classes.templates.TemplateInstance, cognite.client.data_classes.templates.TemplateInstanceList]

Create one or more template instances.

Parameters

- **external_id** (str) – The external id of the template group.
- **version** (int) – The version of the template group to create instances for.
- **instances** (Union[TemplateInstance, Sequence[TemplateInstance]]) – The instances to create.

Returns Created template instance(s).

Return type Union[TemplateInstance, TemplateInstanceList]

Examples

Create new template instances for Covid-19 spread:

```

>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TemplateInstance
>>> c = CogniteClient()
>>> template_instance_1 = TemplateInstance(external_id="norway",
>>>                                       template_name="Country",
>>>                                       field_resolvers={
>>>                                           "name": ConstantResolver("Norway"),
>>>                                           "demographics": ConstantResolver("norway_
↳demographics"),
>>>                                           "deaths": ConstantResolver("Norway_deaths"),
>>>                                           "confirmed": ConstantResolver("Norway_
↳confirmed"),
>>>                                       }
>>>                                       )
>>> template_instance_2 = TemplateInstance(external_id="norway_demographics",
>>>                                       template_name="Demographics",
>>>                                       field_resolvers={
>>>                                           "populationSize": ConstantResolver(5328000),
>>>                                           "growthRate": ConstantResolver(value=0.02)
>>>                                       }
>>>                                       )
>>> c.templates.instances.create("sdk-test-group", 1, [template_instance_1,
↳template_instance_2])

```

Upsert Template instances

TemplateInstancesAPI.**upsert** (*external_id: str, version: int, instances: Union[cognite.client.data_classes.templates.TemplateInstance, Sequence[cognite.client.data_classes.templates.TemplateInstance]]*)
 → Union[cognite.client.data_classes.templates.TemplateInstance, cognite.client.data_classes.templates.TemplateInstanceList]

Upsert one or more template instances. Will overwrite existing instances.

Parameters

- **external_id** (*str*) – The external id of the template group.
- **version** (*int*) – The version of the template group to create instances for.
- **instances** (*Union[TemplateInstance, Sequence[TemplateInstance]]*) – The instances to create.

Returns Created template instance(s).

Return type Union[*TemplateInstance, TemplateInstanceList*]

Examples

Create new template instances for Covid-19 spread:

```

>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TemplateInstance
>>> c = CogniteClient()
>>> template_instance_1 = TemplateInstance(external_id="norway",
>>>                                       template_name="Country",
>>>                                       field_resolvers={

```

(continues on next page)

(continued from previous page)

```

>>>         "name": ConstantResolver("Norway"),
>>>         "demographics": ConstantResolver("norway_demographics"),
>>>         "deaths": ConstantResolver("Norway_deaths"),
>>>         "confirmed": ConstantResolver("Norway_confirmed"),
>>>     }
>>> )
>>> template_instance_2 = TemplateInstance(external_id="norway_demographics",
>>>     template_name="Demographics",
>>>     field_resolvers={
>>>         "populationSize": ConstantResolver(5328000),
>>>         "growthRate": ConstantResolver(0.02)
>>>     }
>>> )
>>> c.templates.instances.upsert("sdk-test-group", 1, [template_instance_1,
↪template_instance_2])

```

Update Template instances

TemplateInstancesAPI.**update** (*external_id*: *str*, *version*: *int*, *item*:
Union[cognite.client.data_classes.templates.TemplateInstanceUpdate,
Sequence[cognite.client.data_classes.templates.TemplateInstanceUpdate]])
→ *Union[cognite.client.data_classes.templates.TemplateInstance,*
cognite.client.data_classes.templates.TemplateInstanceList]

Update one or more template instances :param *item*: Templates instance(s) to update :type *item*:
Union[TemplateInstanceUpdate, Sequence[TemplateInstanceUpdate]]

Returns Updated template instance(s)

Return type *Union[TemplateInstance, TemplateInstanceList]*

Examples

Perform a partial update on a template instance:

```

>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TemplateInstanceUpdate
>>> c = CogniteClient()
>>> my_update = TemplateInstanceUpdate(external_id="test").field_resolvers.add({
↪ "name": ConstantResolver("Norway") })
>>> res = c.templates.instances.update("sdk-test-group", 1, my_update)

```

Retrieve Template instances

TemplateInstancesAPI.**retrieve_multiple** (*external_id*: *str*, *version*: *int*, *ex-*
ternal_ids: *Sequence[str]*, *ig-*
nore_unknown_ids: *bool = False*) → *cog-*
nite.client.data_classes.templates.TemplateInstanceList

Retrieve multiple template instances by external id.

Parameters

- **external_id** (*str*) – The template group to retrieve instances from.
- **version** (*int*) – The version of the template group.

- **external_ids** (*Sequence[str]*) – External IDs of the instances.
- **ignore_unknown_ids** (*bool*) – Ignore external IDs that are not found rather than throw an exception.

Returns The requested template groups.

Return type *TemplateInstanceList*

Examples

Get template instances by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.templates.instances.retrieve_multiple(external_id="sdk-test-group",
↳version=1, external_ids=["abc", "def"])
```

List Template instances

`TemplateInstancesAPI.list` (*external_id: str, version: int, limit: int = 25, data_set_ids: Optional[Sequence[int]] = None, template_names: Optional[Sequence[str]] = None*) → `cognite.client.data_classes.templates.TemplateInstanceList`

Lists instances in a template group. Up to 1000 template instances can be retrieved in one operation.

Parameters

- **external_id** (*str*) – The external id of the template group.
- **version** (*int*) – The version of the template group.
- **limit** (*int*) – Maximum number of template group versions to return. Defaults to 25. Set to -1, float("inf") or None to return all items.
- **data_set_ids** – (Optional[Sequence[int]]): Only include instances which has one of these values in their *data_set_id* field.
- **template_names** – (Optional[Sequence[str]]): Only include instances which has one of these values in their *template_name* field.

Returns List of requested template instances

Return type *TemplateInstanceList*

Examples

List template instances:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> template_instances_list = c.templates.instances.list("template-group-ext-id",
↳1, limit=5)
```

Delete Template instances

TemplateInstancesAPI.**delete** (*external_id: str, version: int, external_ids: Sequence[str], ignore_unknown_ids: bool = False*) → None

Delete one or more template instances.

Parameters

- **external_id** (*str*) – External ID of the template group.
- **version** (*int*) – The version of the template group.
- **external_ids** (*Sequence[str]*) – The external ids of the template instances to delete
- **ignore_unknown_ids** (*bool*) – Ignore external IDs that are not found rather than throw an exception.

Returns None

Examples

Delete template groups by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.templates.instances.delete("sdk-test-group", 1, external_ids=["a", "b"])
```

Create Views

TemplateViewsAPI.**create** (*external_id: str, version: int, views: Union[cognite.client.data_classes.templates.View, Sequence[cognite.client.data_classes.templates.View]]*) → Union[cognite.client.data_classes.templates.View, cognite.client.data_classes.templates.ViewList]

Create one or more template views.

Parameters

- **external_id** (*str*) – The external id of the template group.
- **version** (*int*) – The version of the template group to create views for.
- **views** (*Union[View, Sequence[View]]*) – The views to create.

Returns Created view(s).

Return type Union[View, ViewList]

Examples

Create new views:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes.templates import View, Source
>>> c = CogniteClient()
>>> view = View(external_id="view",
>>>             source=Source(
```

(continues on next page)

(continued from previous page)

```

>>>         type='events',
>>>         filter={
>>>             "startTime": {
>>>                 "min": "$startTime"
>>>             },
>>>             "type": "Test",
>>>         }
>>>         mappings={
>>>             "author": "metadata/author"
>>>         }
>>>     )
>>> c.templates.views.create("sdk-test-group", 1, [view])

```

Upsert Views

TemplateViewsAPI.**upsert** (*external_id*: *str*, *version*: *int*, *views*: *Union[cognite.client.data_classes.templates.View, Sequence[cognite.client.data_classes.templates.View]]*) → *Union[cognite.client.data_classes.templates.View, cognite.client.data_classes.templates.ViewList]*

Upsert one or more template views.

Parameters

- **external_id** (*str*) – The external id of the template group.
- **version** (*int*) – The version of the template group to create views for.
- **views** (*Union[View, Sequence[View]]*) – The views to create.

Returns Created view(s).

Return type *Union[View, ViewList]*

Examples

Upsert new views:

```

>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes.templates import View
>>> c = CogniteClient()
>>> view = View(external_id="view",
>>>             source=Source(
>>>                 type: 'events',
>>>                 filter: {
>>>                     startTime: {
>>>                         min: "$startTime"
>>>                     },
>>>                     type: "Test",
>>>                 }
>>>             mappings: {
>>>                 author: "metadata/author"
>>>             }
>>>         )

```

(continues on next page)

(continued from previous page)

```
>>> )
>>> c.templates.views.upsert("sdk-test-group", 1, [view])
```

List Views

TemplateViewsAPI.**list** (*external_id: str, version: int, limit: int = 25*) → `cognite.client.data_classes.templates.ViewList`
Lists view in a template group. Up to 1000 views can be retrieved in one operation.

Parameters

- **external_id** (*str*) – The external id of the template group.
- **version** (*int*) – The version of the template group.
- **limit** (*int*) – Maximum number of views to return. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns List of requested views

Return type `ViewList`

Examples

List views:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.templates.views.list("template-group-ext-id", 1, limit=5)
```

Resolve View

TemplateViewsAPI.**resolve** (*external_id: str, version: int, view_external_id: str, input: Optional[Dict[str, Any]], limit: int = 25*) → `cognite.client.data_classes.templates.ViewResolveList`

Resolves a View. It resolves the source specified in a View with the provided input and applies the mapping rules to the response.

Parameters

- **external_id** (*str*) – The external id of the template group.
- **version** (*int*) – The version of the template group.
- **input** (*Optional[Dict[str, any]]*) – The input for the View.
- **limit** (*int*) – Maximum number of views to return. Defaults to 25. Set to -1, float("inf") or None to return all items.

Returns The resolved items.

Return type `ViewResolveList`

Examples

Resolve view:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.templates.views.resolve("template-group-ext-id", 1, "view", { "startTime": 10 }, limit=5)
```

Delete Views

TemplateViewsAPI.**delete** (*external_id: str, version: int, view_external_id: Union[Sequence[str], str], ignore_unknown_ids: bool = False*) → None

Delete one or more views.

Parameters

- **external_id** (*str*) – External ID of the template group.
- **version** (*int*) – The version of the template group.
- **view_external_id** (*Union[Sequence[str], str]*) – The external ids of the views to delete
- **ignore_unknown_ids** (*bool*) – Ignore external IDs that are not found rather than throw an exception.

Returns None

Examples

Delete views by external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.templates.views.delete("sdk-test-group", 1, external_id=["a", "b"])
```

Data classes

class cognite.client.data_classes.templates.**ConstantResolver** (*value: Any = None, cognite_client: CogniteClient = None*)

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

Resolves a field to a constant value. The value can be of any supported JSON type.

Parameters **value** (*any*) – The value of the field.

class cognite.client.data_classes.templates.**RawResolver** (*db_name: str = None, table_name: str = None, row_key: str = None, column_name: str = None, cognite_client: CogniteClient = None*)

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

Resolves a field to a RAW column.

Parameters

- **db_name** (*str*) – The database name.
- **table_name** (*str*) – The table name.
- **row_key** (*str*) – The row key.
- **column_name** (*str*) – The column to fetch the value from.

class `cognite.client.data_classes.templates.Source` (*type: str = None, filter: Dict[str, Any] = None, mappings: Dict[str, str] = None, cognite_client: CogniteClient = None*)

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

A source defines the data source with filters and a mapping table.

Parameters

- **type** (*str*) – The type of source. Possible values are: “events”, “assets”, “sequences”, “timeSeries”, “files”.
- **filter** (*Dict[str, any]*) – The filter to apply to the source when resolving the source. A filter also supports binding view input to the filter, by prefixing the input name with ‘\$’.
- **mappings** (*Dict[str, str]*) – The mapping between source result and expected schema.

```

class cognite.client.data_classes.templates.SyntheticTimeSeriesResolver (expression:
    str
    =
    None,
    name:
    Op-
    tional[str]
    =
    None,
    de-
    scrip-
    tion:
    Op-
    tional[str]
    =
    None,
    meta-
    data:
    Op-
    tional[Dict[str,
    str]]
    =
    None,
    is_step:
    Op-
    tional[bool]
    =
    None,
    is_string:
    Op-
    tional[bool]
    =
    None,
    unit:
    Op-
    tional[str]
    =
    None,
    cog-
    nite_client:
    Cog-
    nite-
    Client
    =
    None)

```

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

Resolves a field of type ‘SyntheticTimeSeries’ to a Synthetic Time Series.

Parameters

- **expression** (*str*) – The synthetic time series expression. See this for syntax <https://docs.cognite.com/api/v1/#tag/Synthetic-Time-Series>.
- **name** (*Optional[str]*) – The name of the Time Series.
- **description** (*Optional[str]*) – The description for the Time Series.

- **metadata** (*Optional[Dict[str, str]]*) – Specifies metadata for the Time Series.
- **is_step** (*Optional[bool]*) – Specifies if the synthetic time series is step based.
- **is_string** (*Optional[bool]*) – Specifies if the synthetic time series returned contains string values.
- **unit** (*Optional[str]*) – The unit of the time series.

```
class cognite.client.data_classes.templates.TemplateGroup (external_id: str = None, description: str = None, owners: Optional[List[str]] = None, data_set_id: int = None, created_time: int = None, last_updated_time: int = None, cognite_client: CogniteClient = None)
```

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

A template group is a high level concept encapsulating a schema and a set of template instances. It also has query capability support.

Template groups are versioned, so there can be multiple template groups with the same external ID. The versioning is happening automatically whenever a template groups is changed.

GraphQL schema definition language is used as the language to describe the structure of the templates and data types.

Parameters

- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.
- **description** (*str*) – The description of the template groups.
- **owners** (*List[str]*) – The list of owners for the template groups.
- **data_set_id** (*int*) – The dataSet which this Template Group belongs to

```
class cognite.client.data_classes.templates.TemplateGroupList (resources: Collection[Any], cognite_client: CogniteClient = None)
```

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

```

class cognite.client.data_classes.templates.TemplateGroupVersion (schema: str
                                                                = None,
                                                                version: int
                                                                = None, conflict_mode:
                                                                str =
                                                                None, created_time:
                                                                int = None,
                                                                last_updated_time:
                                                                int =
                                                                None, cognite_client:
                                                                Cognite-
                                                                Client =
                                                                None)

```

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

A Template Group Version supports specifying different conflict modes, which is used when an existing schema already exists.

Patch -> It diff's the new schema with the old schema and fails if there are breaking changes. Update -> It sets the new schema as schema of a new version. Force -> It ignores breaking changes and replaces the old schema with the new schema. The default mode is "patch".

Parameters

- **schema** (*str*) – The GraphQL schema.
- **version** (*int*) – Incremented by the server whenever the schema of a template groups changes.
- **conflict_mode** (*str*) – Can be set to 'Patch', 'Update' or 'Force'.

```

class cognite.client.data_classes.templates.TemplateGroupVersionList (resources:
                                                                Collec-
                                                                tion[Any],
                                                                cog-
                                                                nite_client:
                                                                Cog-
                                                                nite-
                                                                Client
                                                                =
                                                                None)

```

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

```
class cognite.client.data_classes.templates.TemplateInstance (external_id: str
                                                         = None, tem-
                                                         plate_name:
                                                         str = None,
                                                         field_resolvers:
                                                         Dict[str, FieldRe-
                                                         solvers] = None,
                                                         data_set_id: Op-
                                                         tional[int] = None,
                                                         created_time:
                                                         int = None,
                                                         last_updated_time:
                                                         int = None,
                                                         cognite_client:
                                                         CogniteClient =
                                                         None)
```

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

A template instance that implements a template by specifying a resolver per field.

Parameters

- **external_id** (*str*) – The id of the template instance.
- **template_name** (*str*) – The template name to implement.
- **field_resolvers** (*Dict[str, FieldResolvers]*) – A set of field resolvers where the dictionary key correspond to the field name.
- **data_set_id** (*int*) – The id of the dataset this instance belongs to.
- **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.

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]

```
class cognite.client.data_classes.templates.TemplateInstanceList (resources:
                                                         Collec-
                                                         tion[Any],
                                                         cog-
                                                         nite_client:
                                                         Cognite-
                                                         Client =
                                                         None)
```

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

```
class cognite.client.data_classes.templates.TemplateInstanceUpdate (id: Optional[int]
                                                                = None,
                                                                external_id: Optional[str]
                                                                = None)
```

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

Changes applied to template instance

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

```
class cognite.client.data_classes.templates.View (external_id: str = None, source: Source = None, data_set_id: Optional[int] = None, created_time: int = None, last_updated_time: int = None, cognite_client: CogniteClient = None)
```

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

A view is used to map existing data to a type in the template group. A view supports input, that can be bound to the underlying filter.

Parameters

- **external_id** (*str*) – The external ID provided by the client. Must be unique for the resource type.
- **source** (*Source*) – Defines the data source for the view.
- **data_set_id** (*Optional[int]*) – The dataSetId of the view

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]

```
class cognite.client.data_classes.templates.ViewList (resources: Collection[Any], cognite_client: CogniteClient = None)
```

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

```
class cognite.client.data_classes.templates.ViewResolveList (resources: Collection[Any], cognite_client: CogniteClient = None)
```

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

```
class cognite.client.data_classes.templates.ViewResolver (external_id: str = None, input: Optional[Dict[str, Any]] = None, cognite_client: CogniteClient = None)
```

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

Resolves the field by loading the data from a view.

Parameters

- **external_id** (*str*) – The external id of the view.
- **input** (*Optional[Dict[str, any]]*) – The input used to resolve the view.

2.4.19 Annotations

Retrieve an annotation by id

`AnnotationsAPI.retrieve(id: int) → Optional[cognite.client.data_classes.annotations.Annotation]`

Retrieve an annotation by id

Parameters **id** (*int*) – id of the annotation to be retrieved

Returns annotation requested

Return type *Annotation*

Retrieve multiple annotations by id

`AnnotationsAPI.retrieve_multiple(ids: Sequence[int]) → cognite.client.data_classes.annotations.AnnotationList`

Retrieve annotations by IDs

Parameters (**Sequence[int]**) (*ids*) – list of IDs to be retrieved

Returns list of annotations

Return type *AnnotationList*

List annotation

`AnnotationsAPI.list(filter: Union[cognite.client.data_classes.annotations.AnnotationFilter, Dict[KT, VT]], limit: int = 25) → cognite.client.data_classes.annotations.AnnotationList`

List annotations.

Parameters

- **limit** (*int*) – Maximum number of annotations to return. Defaults to 25.
- **filter** (*AnnotationFilter*) – Return annotations with parameter values that matches what is specified. Note that `annotated_resource_type` and `annotated_resource_ids` are always required.

Returns list of annotations

Return type *AnnotationList*

Create an annotation

`AnnotationsAPI.create(annotations: Union[cognite.client.data_classes.annotations.Annotation, Sequence[cognite.client.data_classes.annotations.Annotation]]) → Union[cognite.client.data_classes.annotations.Annotation, cognite.client.data_classes.annotations.AnnotationList]`

Create annotations

Parameters `annotations` (`Union[Annotation, Sequence[Annotation]]`) – annotation(s) to create

Returns created annotation(s)

Return type `Union[Annotation, AnnotationList]`

Suggest an annotation

`AnnotationsAPI.suggest` (`annotations: Union[cognite.client.data_classes.annotations.Annotation, Sequence[cognite.client.data_classes.annotations.Annotation]]`)
 → `Union[cognite.client.data_classes.annotations.Annotation, cognite.client.data_classes.annotations.AnnotationList]`

Suggest annotations

Parameters `annotations` (`Union[Annotation, Sequence[Annotation]]`) – annotation(s) to suggest. They must have status set to “suggested”.

Returns suggested annotation(s)

Return type `Union[Annotation, AnnotationList]`

Update annotations

`AnnotationsAPI.update` (`item: Union[cognite.client.data_classes.annotations.Annotation, cognite.client.data_classes.annotations.AnnotationUpdate, Sequence[Union[cognite.client.data_classes.annotations.Annotation, cognite.client.data_classes.annotations.AnnotationUpdate]]]`)
 → `Union[cognite.client.data_classes.annotations.Annotation, cognite.client.data_classes.annotations.AnnotationList]`

Update annotations

Parameters `item` (`Union[Annotation, AnnotationUpdate, Sequence[Union[Annotation, AnnotationUpdate]]]`) – Annotation or list of annotations to update (or patch or list of patches to apply)

Delete annotations

`AnnotationsAPI.delete` (`id: Union[int, Sequence[int]]`) → None

Delete annotations

Parameters `id` (`Union[int, Sequence[int]]`) – ID or list of IDs to be deleted

Data classes

```
class cognite.client.data_classes.annotations.Annotation (annotation_type: str;
data: dict, status: str;
creating_app: str, creating_app_version:
str, creating_user:
Optional[str], annotated_resource_type: str;
annotated_resource_id:
Optional[int] = None)
```

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

Representation of an annotation in CDF.

Parameters

- **annotation_type** (*str*) – The type of the annotation. This uniquely decides what the structure of the ‘data’ block will be.
- **data** (*dict*) – The annotation information. The format of this object is decided by and validated against the ‘annotation_type’ attribute.
- **status** (*str*) – The status of the annotation, e.g. “suggested”, “approved”, “rejected”.
- **annotated_resource_type** (*str*) – Type name of the CDF resource that is annotated, e.g. “file”.
- **annotated_resource_id** (*int, optional*) – The internal ID of the annotated resource.
- **creating_app** (*str*) – The name of the app from which this annotation was created.
- **creating_app_version** (*str*) – The version of the app that created this annotation. Must be a valid semantic versioning (SemVer) string.
- **creating_user** – (*str, optional*): A username, or email, or name. This is not checked nor enforced. If the value is None, it means the annotation was created by a service.
- **id** (*int, optional*) – A server-generated id for the object. Read-only.
- **created_time** (*int, optional*) – Time when this annotation was created in CDF. The time is measured in milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds. Read-only.
- **last_updated_time** (*int, optional*) – Time when this annotation was last updated in CDF. The time is measured in milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds. Read-only.
- **cognite_client** (*CogniteClient, optional*) – The client to associate with this object. Read-only.

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]

```
class cognite.client.data_classes.annotations.AnnotationFilter (annotated_resource_type:
                                                                    str,
                                                                    anno-
                                                                    tated_resource_ids:
                                                                    List[Dict[str,
                                                                    int]],
                                                                    status:
                                                                    Optional[str]
                                                                    = None,
                                                                    cre-
                                                                    ating_user:
                                                                    Optional[str] =
                                                                    ”,
                                                                    creating_app:
                                                                    Optional[str]
                                                                    = None,
                                                                    creat-
                                                                    ing_app_version:
                                                                    Optional[str]
                                                                    = None,
                                                                    an-
                                                                    notation_type:
                                                                    Optional[str] =
                                                                    None,
                                                                    data:
                                                                    Optional[Dict[str,
                                                                    Any]] = None)
```

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

Filter on annotations with various criteria

Parameters

- **annotated_resource_type** (*str*) – The type of the CDF resource that is annotated, e.g. “file”.
- **annotated_resource_ids** (*List[Dict[str, Any]]*) – List of ids of the annotated CDF resources to filter in. Example format: [{"id": 1234}, {"id": “4567”}]. Must contain at least one item.
- **status** (*str, optional*) – Status of annotations to filter for, e.g. “suggested”, “approved”, “rejected”.
- **creating_user** (*str, optional*) – Name of the user who created the annotations to filter for. Can be set explicitly to “None” to filter for annotations created by a service.
- **creating_app** (*str, optional*) – Name of the app from which the annotations to filter for were created.
- **creating_app_version** (*str, optional*) – Version of the app from which the annotations to filter for were created.
- **annotation_type** (*str, optional*) – Type name of the annotations.
- **data** (*Dict[str, Any], optional*) – The annotation data to filter by. Example format: {"label": “cat”, “confidence”: 0.9}

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]`

```
class cognite.client.data_classes.annotations.AnnotationList (resources: Collection[Any],  
cognite_client: CogniteClient = None)  
    Bases: cognite.client.data_classes._base.CogniteResourceList
```

```
class cognite.client.data_classes.annotations.AnnotationUpdate (id: int)  
    Bases: cognite.client.data_classes._base.CogniteUpdate
```

Changes applied to annotation

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

2.4.20 Identity and access management

Tokens

Inspect the token currently used by the client

`TokenAPI.inspect()` → `cognite.client.data_classes.iam.TokenInspection`
Inspect a token.

Get details about which projects it belongs to and which capabilities are granted to it.

Returns The object with token inspection details.

Return type *TokenInspection*

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, Sequence[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, Sequence[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, Sequence[int]]`) → `None`

Delete one or more service accounts.

Parameters `id` (`Union[int, Sequence[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: Optional[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, Sequence[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, Sequence[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, Sequence[int]]*) → None

Delete one or more api keys.

Parameters `id` (*Union[int, Sequence[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, Sequence[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, Sequence[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, Sequence[int]]*) → None

Delete one or more groups.

Parameters **id** (*Union[int, Sequence[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, Sequence[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, Sequence[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, Sequence[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, Sequence[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, Sequence[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, Sequence[int]]*) → None

Delete one or more security categories.

Parameters *id* (*Union[int, Sequence[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)
```

Sessions

List sessions

`SessionsAPI.list` (*status: Optional[str] = None*) → `cognite.client.data_classes.iam.SessionList`

List all sessions in the current project.

Parameters *status* (*Optional[str]*) – If given, only sessions with the given status are returned.

Returns a list of sessions in the current project.

Return type *SessionList*

Create a session

`SessionsAPI.create` (*client_credentials: Optional[cognite.client.data_classes.iam.ClientCredentials] = None*) → `cognite.client.data_classes.iam.CreatedSession`

Create a session.

Parameters `client_credentials` (*Optional*[*ClientCredentials*]) – The client credentials to create the session. If set to None, a session will be created using the credentials used to instantiate this *CogniteClient* object. If that was done using a token, a session will be created using token exchange. Similarly, if the credentials were client credentials, a session will be created using client credentials. This method does not work when using client certificates (not supported server-side).

Returns The object with token inspection details.

Return type *CreatedSession*

Revoke a session

`SessionsAPI.revoke` (*id*: *Union*[*int*, *Sequence*[*int*]]) → *cognite.client.data_classes.iam.SessionList*
 Revoke access to a session. Revocation of a session may in some cases take up to 1 hour to take effect.

Parameters `id` (*Union*[*int*, *Sequence*[*int*]]) – Id or list of session ids

Returns List of revoked sessions. If the user does not have the `sessionsAcl:LIST` capability, then only the session IDs will be present in the response.

Return type *SessionList*

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:
                                             CogniteClient = 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: Collection[Any], cog-
                                             nite_client: CogniteClient = None)
```

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

```
class cognite.client.data_classes.iam.ClientCredentials (client_id: str,
                                                         client_secret: str)
```

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

Client credentials for session creation

Parameters

- `client_id` (*str*) – Client ID from identity provider.
- `client_secret` (*str*) – Client secret from identity provider.

```
class cognite.client.data_classes.iam.CreatedSession (id: int = None, type: str  
                                                    = None, status: str = None,  
                                                    nonce: str = None, client_id:  
                                                    str = None, cognite_client:  
                                                    CogniteClient = None)
```

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

Session creation related information

Parameters

- **id** (*int*) – ID of the created session.
- **type** (*str*) – Credentials kind used to create the session.
- **status** (*str*) – Current status of the session.
- **nonce** (*str*) – Nonce to be passed to the internal service that will bind the session
- **client_id** (*str*) – Client ID in identity provider. Returned only if the session was created using client credentials

```
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:  
                                             CogniteClient = 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: Collection[Any], cog-  
                                                nite_client: CogniteClient = None)
```

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

```
class cognite.client.data_classes.iam.ProjectSpec (url_name: str, groups: List[int])
```

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

A CDF project spec

Parameters

- **url_name** (*str*) – The url name for the project
- **groups** (*List[int]*) – Group ids in the project

```
class cognite.client.data_classes.iam.SecurityCategory (name: str = None, id: int =  
                                                         None, cognite_client: Cog-  
                                                         niteClient = 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:      Col-
                                                                    lection[Any],      cog-
                                                                    nite_client:      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, cog-
                                                                    nite_client:  CogniteClient =
                                                                    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:      Col-
                                                                    lection[Any],      cog-
                                                                    nite_client:      Cognite-
                                                                    Client = None)
```

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

```
class cognite.client.data_classes.iam.Session (id:  int = None, type:  str = None, status:
                                                                    str = None, creation_time:  int = None, ex-
                                                                    piration_time:  int = None, client_id:  str
                                                                    = None, cognite_client:  CogniteClient =
                                                                    None)
```

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

Session status

Parameters

- **id** (*int*) – ID of the session.
- **type** (*str*) – Credentials kind used to create the session.
- **status** (*str*) – Current status of the session.
- **creation_time** (*int*) – Session creation time, in milliseconds since 1970

- **expiration_time** (*int*) – Session expiry time, in milliseconds since 1970. This value is updated on refreshing a token
- **client_id** (*str*) – Client ID in identity provider. Returned only if the session was created using client credentials

```
class cognite.client.data_classes.iam.SessionList (resources: Collection[Any], cognite_client: CogniteClient = None)
    Bases: cognite.client.data_classes._base.CogniteResourceList
```

```
class cognite.client.data_classes.iam.TokenInspection (subject: str, projects: List[cognite.client.data_classes.iam.ProjectSpec],
    capabilities: List[Dict[KT, VT]])
    Bases: cognite.client.data_classes._base.CogniteResponse
```

Current login status

Parameters

- **subject** (*str*) – Subject (sub claim) of JWT.
- **projects** (*List[ProjectSpec]*) – Projects this token is valid for.
- **capabilities** (*List[Dict]*) – Capabilities associated with this token.

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]

2.4.21 Extraction pipelines

List extraction pipelines

ExtractionPipelinesAPI.**list** (*limit: int = 25*) → cognite.client.data_classes.extractionpipelines.ExtractionPipelineList
List extraction pipelines

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

Returns List of requested ExtractionPipelines

Return type *ExtractionPipelineList*

Examples

List ExtractionPipelines:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> ep_list = c.extraction_pipelines.list(limit=5)
```

Create extraction pipeline

`ExtractionPipelinesAPI.create` (*extraction_pipeline: Union[cognite.client.data_classes.extractionpipelines.ExtractionPipeline, Sequence[cognite.client.data_classes.extractionpipelines.ExtractionPipeline], None]* = *None*, ***kwargs*) → *Union[cognite.client.data_classes.extractionpipelines.ExtractionPipeline, cognite.client.data_classes.extractionpipelines.ExtractionPipelineList]*

Create one or more extraction pipelines.

You can create an arbitrary number of extraction pipelines, and the SDK will split the request into multiple requests if necessary.

Parameters `extraction_pipeline` (*Union[ExtractionPipeline, List[ExtractionPipeline]]*) – Extraction pipeline or list of extraction pipelines to create.

Returns Created extraction pipeline(s)

Return type *Union[ExtractionPipeline, ExtractionPipelineList]*

Examples

Create new extraction pipeline:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import ExtractionPipeline
>>> c = CogniteClient()
>>> extpipes = [ExtractionPipeline(name="extPipe1", ...), ExtractionPipeline(name=
↳"extPipe2", ...)]
>>> res = c.extraction_pipelines.create(extpipes)
```

Retrieve an extraction pipeline by ID

`ExtractionPipelinesAPI.retrieve` (*id: Optional[int]* = *None*, *external_id: Optional[str]* = *None*) → *Optional[cognite.client.data_classes.extractionpipelines.ExtractionPipeline]*

Retrieve a single extraction pipeline by id.

Parameters

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

Returns Requested extraction pipeline or None if it does not exist.

Return type *Optional[ExtractionPipeline]*

Examples

Get extraction pipeline by id:

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

Get extraction pipeline by external id:

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

Retrieve multiple extraction pipelines by ID

ExtractionPipelinesAPI.**retrieve_multiple**(*ids: Optional[Sequence[int]] = None, external_ids: Optional[Sequence[str]] = None, ignore_unknown_ids: bool = False*) → `cognite.client.data_classes.extractionpipelines.ExtractionPipelineList`

Retrieve multiple extraction pipelines by ids and external ids.

Parameters

- **ids** (*Sequence[int], optional*) – IDs
- **external_ids** (*Sequence[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 ExtractionPipelines.

Return type `ExtractionPipelineList`

Examples

Get ExtractionPipelines by id:

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

Get assets by external id:

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

Update extraction pipelines

ExtractionPipelinesAPI.**update**(*item: Union[cognite.client.data_classes.extractionpipelines.ExtractionPipeline, cognite.client.data_classes.extractionpipelines.ExtractionPipelineUpdate, Sequence[Union[cognite.client.data_classes.extractionpipelines.ExtractionPipeline, cognite.client.data_classes.extractionpipelines.ExtractionPipelineUpdate]]]*) → `Union[cognite.client.data_classes.extractionpipelines.ExtractionPipeline, cognite.client.data_classes.extractionpipelines.ExtractionPipelineList]`

Update one or more extraction pipelines

Parameters **item** (`Union[ExtractionPipeline, ExtractionPipelineUpdate, Sequence[Union[ExtractionPipeline, ExtractionPipelineUpdate]]]`) – Extraction pipeline(s) to update

Returns Updated extraction pipeline(s)

Return type Union[*ExtractionPipeline*, *ExtractionPipelineList*]

Examples

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

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> update = ExtractionPipelineUpdate(id=1)
>>> update.description.set("Another new extpipe")
>>> res = c.extraction_pipelines.update(update)
```

Delete extraction pipelines

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

Delete one or more extraction pipelines

Parameters

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

Returns None

Examples

Delete extraction pipelines by id or external id:

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

Extraction pipeline runs

List runs for an extraction pipeline

`ExtractionPipelineRunsAPI.list` (*external_id*: str, *statuses*: Optional[Sequence[str]] = None, *message_substring*: Optional[str] = None, *created_time*: Union[Dict[str, Any], cognite.client.data_classes.shared.TimestampRange, None] = None, *limit*: int = 25) → cognite.client.data_classes.extractionpipelines.ExtractionPipelineRunList

List runs for an extraction pipeline with given `external_id`

Parameters

- **external_id** (str) – Extraction pipeline external Id.
- **statuses** (Sequence[str]) – One or more among “success” / “failure” / “seen”.
- **message_substring** (str) – Failure message part.
- **created_time** (int) – The number of milliseconds since 00:00:00 Thursday, 1 January 1970, Coordinated Universal Time (UTC), minus leap seconds.

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

Returns List of requested extraction pipeline runs

Return type *ExtractionPipelineRunList*

Examples

List extraction pipeline runs:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> runsList = c.extraction_pipelines.runs.list(external_id="test ext id",
↳limit=5)
```

Filter extraction pipeline runs on a given status:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> runsList = c.extraction_pipelines.runs.list(external_id="test ext id",
↳statuses=["seen"], statuslimit=5)
```

Report new runs

`ExtractionPipelineRunsAPI.create` (*run*: *Union[cognite.client.data_classes.extractionpipelines.ExtractionPipelineRun, Sequence[cognite.client.data_classes.extractionpipelines.ExtractionPipelineRun]]*)
→ *Union[cognite.client.data_classes.extractionpipelines.ExtractionPipelineRun, cognite.client.data_classes.extractionpipelines.ExtractionPipelineRunList]*

Create one or more extraction pipeline runs.

You can create an arbitrary number of extraction pipeline runs, and the SDK will split the request into multiple requests.

Parameters `run` (*Union[ExtractionPipelineRun, Sequence[ExtractionPipelineRun]]*)
– Extraction pipeline or list of extraction pipeline runs to create.

Returns Created extraction pipeline run(s)

Return type *Union[ExtractionPipelineRun, ExtractionPipelineRunList]*

Examples

Report a new extraction pipeline run:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import ExtractionPipelineRun
>>> c = CogniteClient()
>>> res = c.extraction_pipelines.runs.create(
...     ExtractionPipelineRun(status="success", extpipe_external_id="extId"))
```

Extraction pipeline configs

Get the latest or a specific config revision

ExtractionPipelineConfigsAPI.**retrieve**(*external_id*: str, *revision*: Optional[int] = None, *active_at_time*: Optional[int] = None) → cognite.client.data_classes.extractionpipelines.ExtractionPipelineConfig

Retrieve a specific configuration revision, or the latest by default
<<https://docs.cognite.com/api/v1/#tag/Extraction-Pipelines-Config/operation/getExtPipeConfigRevision>>

By default the latest configuration revision is retrieved, or you can specify a timestamp or a revision number.

Parameters

- **external_id** (str) – External id of the extraction pipeline to retrieve config from.
- **revision** (Optional[int]) – Optionally specify a revision number to retrieve.
- **active_at_time** (Optional[int]) – Optionally specify a timestamp the configuration revision should be active.

Returns Retrieved extraction pipeline configuration revision

Return type *ExtractionPipelineConfig*

Examples

Retrieve latest config revision:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.extraction_pipelines.config.retrieve("extId")
```

List configuration revisions

ExtractionPipelineConfigsAPI.**list**(*external_id*: str) → cognite.client.data_classes.extractionpipelines.ExtractionPipelineConfigRevisionList

Retrieve all configuration revisions from an extraction pipeline <<https://docs.cognite.com/api/v1/#tag/Extraction-Pipelines-Config/operation/listExtPipeConfigRevisions>>

Parameters **external_id** (str) – External id of the extraction pipeline to retrieve config from.

Returns Retrieved extraction pipeline configuration revisions

Return type *ExtractionPipelineConfigRevisionList*

Examples

Retrieve a list of config revisions:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.extraction_pipelines.config.list("extId")
```

Create a config revision

ExtractionPipelineConfigsAPI.**create** (*config: cognite.client.data_classes.extractionpipelines.ExtractionPipelineConfig*)
→ *cognite.client.data_classes.extractionpipelines.ExtractionPipelineConfig*
Create a new configuration revision <<https://docs.cognite.com/api/v1/#tag/Extraction-Pipelines-Config/operation/createExtPipeConfig>>

Parameters **config** (*ExtractionPipelineConfig*) – Configuration revision to create.

Returns Created extraction pipeline configuration revision

Return type *ExtractionPipelineConfig*

Examples

Create a config revision:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.config import ExtractionPipelineConfig
>>> c = CogniteClient()
>>> res = c.extraction_pipelines.config.create(ExtractionPipelineConfig(external_
↪id="extId", config="my config contents"))
```

Revert to an earlier config revision

ExtractionPipelineConfigsAPI.**revert** (*external_id: str, revision: int*) → *cognite.client.data_classes.extractionpipelines.ExtractionPipelineConfig*
Revert to a previous configuration revision <<https://docs.cognite.com/api/v1/#tag/Extraction-Pipelines-Config/operation/createExtPipeConfig>>

Parameters

- **external_id** (*str*) – External id of the extraction pipeline to revert revision for.
- **revision** (*int*) – Revision to revert to.

Returns New latest extraction pipeline configuration revision.

Return type *ExtractionPipelineConfig*

Examples

Revert a config revision:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.extraction_pipelines.config.revert("extId", 5)
```

Data classes

```

class cognite.client.data_classes.extractionpipelines.ExtractionPipeline (id:
    int
    =
    None,
    ex-
    ter-
    nal_id:
    str
    =
    None,
    name:
    str
    =
    None,
    de-
    scrip-
    tion:
    str
    =
    None,
    data_set_id:
    int
    =
    None,
    raw_tables:
    List[Dict[str,
    str]]
    =
    None,
    last_success:
    int
    =
    None,
    last_failure:
    int
    =
    None,
    last_message:
    str
    =
    None,
    last_seen:
    int
    =
    None,
    sched-
    ule:
    str
    =
    None,
    con-
    tacts:
    List[ExtractionPipelineCo
    =
    None,
    meta-
    data:
    Dict[str,

```

An extraction pipeline is a representation of a process writing data to CDF, such as an extractor or an ETL tool.

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.
- **name** (*str*) – The name of the extraction pipeline.
- **description** (*str*) – The description of the extraction pipeline.
- **data_set_id** (*int*) – The id of the dataset this extraction pipeline related with.
- **raw_tables** (*List[Dict[str, str]]*) – list of raw tables in list format: [{"db-Name": "value", "tableName": "value"}].
- **last_success** (*int*) – Milliseconds value of last success status.
- **last_failure** (*int*) – Milliseconds value of last failure status.
- **last_message** (*str*) – Message of last failure.
- **last_seen** (*int*) – Milliseconds value of last seen status.
- **schedule** (*str*) – None/On trigger/Continuous/cron regex.
- **contacts** (*List[ExtractionPipelineContact]*) – list of contacts
- **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*) – Source text value for extraction pipeline.
- **documentation** (*str*) – Documentation text value for extraction pipeline.
- **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.
- **created_by** (*str*) – Extraction pipeline creator, usually an email.
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

```

class cognite.client.data_classes.extractionpipelines.ExtractionPipelineConfig(external_id:
    str
    =
    None,
    config:
    str
    =
    None,
    re-
    vi-
    sion:
    int
    =
    None,
    de-
    scrip-
    tion:
    str
    =
    None,
    cre-
    ated_time:
    int
    =
    None,
    cog-
    nite_client:
    Cog-
    nite-
    Client
    =
    None)

```

Bases: `cognite.client.data_classes.extractionpipelines.ExtractionPipelineConfigRevision`

An extraction pipeline config

Parameters

- **external_id** (*str*) – The external ID of the associated extraction pipeline.
- **config** (*str*) – Contents of this configuration revision.
- **revision** (*int*) – The revision number of this config as a positive integer.
- **description** (*str*) – Short description of this configuration revision.
- **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.extractionpipelines.ExtractionPipelineConfigRevision (exter
    str
    =
    None
    re-
    vi-
    sion:
    int
    =
    None
    de-
    scrip-
    tion:
    str
    =
    None
    cre-
    ated_
    int
    =
    None
    cog-
    nite_
    Cog-
    nite-
    Clie
    =
    None
```

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

An extraction pipeline config revision

Parameters

- **external_id** (*str*) – The external ID of the associated extraction pipeline.
- **revision** (*int*) – The revision number of this config as a positive integer.
- **description** (*str*) – Short description of this configuration revision.
- **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.extractionpipelines.ExtractionPipelineConfigRevisionList
```

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

```
class cognite.client.data_classes.extractionpipelines.ExtractionPipelineContact (name:
                                                                    str,
                                                                    email:
                                                                    str,
                                                                    role:
                                                                    str,
                                                                    send_notifications:
                                                                    bool)
```

Bases: dict

A contact for an extraction pipeline

Parameters

- **name** (*str*) – Name of contact
- **email** (*str*) – Email address of contact
- **role** (*str*) – Role of contact, such as Owner, Maintainer, etc.
- **send_notifications** (*bool*) – Whether to send notifications to this contact or not

```
class cognite.client.data_classes.extractionpipelines.ExtractionPipelineList (resources:
                                                                    Col-
                                                                    lec-
                                                                    tion[Any],
                                                                    cog-
                                                                    nite_client:
                                                                    Cog-
                                                                    nite-
                                                                    Client
                                                                    =
                                                                    None)
```

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

```
class cognite.client.data_classes.extractionpipelines.ExtractionPipelineRun (extpipe_external_id:  
                                                                    str  
                                                                    =  
                                                                    None,  
                                                                    sta-  
                                                                    tus:  
                                                                    str  
                                                                    =  
                                                                    None,  
                                                                    mes-  
                                                                    sage:  
                                                                    str  
                                                                    =  
                                                                    None,  
                                                                    cre-  
                                                                    ated_time:  
                                                                    int  
                                                                    =  
                                                                    None,  
                                                                    cog-  
                                                                    nite_client:  
                                                                    Cog-  
                                                                    nite-  
                                                                    Client  
                                                                    =  
                                                                    None,  
                                                                    id:  
                                                                    int  
                                                                    =  
                                                                    None,  
                                                                    **kwargs)
```

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

A representation of an extraction pipeline run.

Parameters

- **id** (*int*) – A server-generated ID for the object.
- **extpipe_external_id** (*str*) – The external ID of the extraction pipeline.
- **status** (*str*) – success/failure/seen.
- **message** (*str*) – Optional status message.
- **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.

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]

```

class cognite.client.data_classes.extractionpipelines.ExtractionPipelineRunFilter (external_id:
    str
    =
    None,
    status:
    Sequence[str]
    =
    None,
    message:
    StringFilter
    =
    None,
    created_time:
    Union[Dict[
    Any],
    TimestampRange]
    =
    None,
    cognite_client:
    CogniteClient
    =
    None)

```

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

Filter runs with exact matching

Parameters

- **external_id** (*str*) – The external ID of related ExtractionPipeline provided by the client. Must be unique for the resource type.
- **status** (*Sequence[str]*) – success/failure/seen.
- **message** (*StringFilter*) – message filter.
- **created_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.extractionpipelines.ExtractionPipelineRunList (resources:  
    Collection[Any],  
    cognite_client:  
    CogniteClient  
    =  
    None)
```

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

```
class cognite.client.data_classes.extractionpipelines.ExtractionPipelineUpdate (id:  
    Optional[int]  
    =  
    None,  
    external_id:  
    Optional[str]  
    =  
    None)
```

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

Changes applied to an extraction pipeline

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.

```
class cognite.client.data_classes.extractionpipelines.StringFilter (substring:  
    Optional[str]  
    = None)
```

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

Filter runs on substrings of the message

Parameters **substring** (*str*) – Part of message

2.4.22 Transformations

Create transformations

```
TransformationsAPI.create (transformation: Union[cognite.client.data_classes.transformations.Transformation,  
    Sequence[cognite.client.data_classes.transformations.Transformation]])  
    → Union[cognite.client.data_classes.transformations.Transformation,  
    cognite.client.data_classes.transformations.TransformationList]
```

Create one or more transformations.

Parameters **transformation** (*Union[Transformation,*
 List[Transformation]]) – Transformation or list of transformations to create.

Returns Created transformation(s)

Examples

Create new transformations:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import Transformation, TransformationDestination
↳TransformationDestination
>>> from cognite.client.data_classes.transformations.common import ViewInfo, EdgeType
↳EdgeType
>>> c = CogniteClient()
>>> transformations = [
>>>     Transformation(
>>>         name="transformation1",
>>>         destination=TransformationDestination.assets()
>>>     ),
>>>     Transformation(
>>>         name="transformation2",
>>>         destination=TransformationDestination.raw("myDatabase", "myTable")
>>>     ),
>>>     Transformation(
>>>         name="transformation3",
>>>         view = ViewInfo(space="TypeSpace", external_id="TypeExtId", version="version"),
↳"version"),
>>>         destination=TransformationDestination.instance_nodes(view, "InstanceSpace")
↳")
>>>     ),
>>>     Transformation(
>>>         name="transformation4",
>>>         view = ViewInfo(space="TypeSpace", external_id="TypeExtId", version="version"),
↳"version"),
>>>         destination=TransformationDestination.instance_edges(view, "InstanceSpace")
↳")
>>>     ),
>>>     Transformation(
>>>         name="transformation5",
>>>         edge_type = EdgeType(space="TypeSpace", external_id="TypeExtId"),
>>>         destination=TransformationDestination.instance_edges(edge_type, "InstanceSpace")
↳"InstanceSpace")
>>>     ),
>>> ]
>>> res = c.transformations.create(transformations)
```

Retrieve transformations by id

TransformationsAPI.**retrieve** (*id: Optional[int] = None, external_id: Optional[str] = None*) → Optional[cognite.client.data_classes.transformations.Transformation]
 Retrieve a single transformation by id.

Parameters *id* (*int, optional*) – ID

Returns Requested transformation or None if it does not exist.

Return type Optional[Transformation]

Examples

Get transformation by id:

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

Get transformation by external id:

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

TransformationsAPI.**retrieve_multiple** (*ids: Optional[Sequence[int]] = None, external_ids: Optional[Sequence[str]] = None, ignore_unknown_ids: bool = False*) → *cognite.client.data_classes.transformations.TransformationList*

Retrieve multiple transformations.

Parameters

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

Returns Requested transformation or None if it does not exist.

Return type *TransformationList*

Examples

Get multiple transformations:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.transformations.retrieve_multiple(ids=[1,2,3], external_ids=[
↪ 'transform-1', 'transform-2'])
```

Run transformations by id

TransformationsAPI.**run** (*transformation_id: Optional[int] = None, transformation_external_id: Optional[str] = None, wait: bool = True, timeout: Optional[float] = None*) → *cognite.client.data_classes.transformations.jobs.TransformationJob*

Run a transformation.

Parameters

- **transformation_id** (*int*) – Transformation internal id
- **transformation_external_id** (*str*) – Transformation external id
- **wait** (*bool*) – Wait until the transformation run is finished. Defaults to True.
- **timeout** (*Optional[float]*) – maximum time (s) to wait, default is None (infinite time). Once the timeout is reached, it returns with the current status. Won't have any effect if wait is False.

Returns Created transformation job

Examples

Run transformation to completion by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>>
>>> res = c.transformations.run(transformation_id = 1)
```

Start running transformation by id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>>
>>> res = c.transformations.run(transformation_id = 1, wait = False)
```

`TransformationsAPI.run_async` (*transformation_id*: *Optional[int]* = *None*, *transformation_external_id*: *Optional[str]* = *None*, *timeout*: *Optional[float]* = *None*) → `cognite.client.data_classes.transformations.jobs.TransformationJob`

Run a transformation to completion asynchronously.

Parameters

- **transformation_id** (*int*) – internal Transformation id
- **transformation_external_id** (*str*) – external Transformation id
- **timeout** (*Optional[float]*) – maximum time (s) to wait, default is None (infinite time). Once the timeout is reached, it returns with the current status.

Returns Completed (if finished) or running (if timeout reached) transformation job.

Examples

Run transformation asynchronously by id:

```
>>> import asyncio
>>> from cognite.client import CogniteClient
>>>
>>> c = CogniteClient()
>>>
>>> async def run_transformation():
>>>     res = await c.transformations.run_async(id = 1)
>>>
>>> loop = asyncio.get_event_loop()
>>> loop.run_until_complete(run_transformation())
>>> loop.close()
```

Preview transformations

`TransformationsAPI.preview` (*query*: *Optional[str]* = *None*, *convert_to_string*: *bool* = *False*, *limit*: *int* = *100*, *source_limit*: *Optional[int]* = *100*, *infer_schema_limit*: *Optional[int]* = *1000*) → `cognite.client.data_classes.transformations.TransformationPreviewResult`

Preview the result of a query.

Parameters

- **query** (*str*) – SQL query to run for preview.
- **convert_to_string** (*bool*) – Stringify values in the query results, default is False.
- **limit** (*int*) – Maximum number of rows to return in the final result, default is 100.
- **source_limit** (*Union[int, str]*) – Maximum number of items to read from the data source or None to run without limit, default is 100.
- **infer_schema_limit** – Limit for how many rows that are used for inferring result schema, default is 1000.

Returns Result of the executed query

Examples

Preview transformation results as schema and list of rows:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>>
>>> query_result = c.transformations.preview(query="select * from _cdf.assets")
```

Preview transformation results as pandas dataframe:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>>
>>> df = c.transformations.preview(query="select * from _cdf.assets").to_pandas()
```

Cancel transformation run by id

TransformationsAPI.**cancel** (*transformation_id: Optional[int] = None, transformation_external_id: Optional[str] = None*) → None

Cancel a running transformation.

Parameters

- **transformation_id** (*int*) – Transformation internal id
- **transformation_external_id** (*str*) – Transformation external id

Examples

Wait transformation for 1 minute and cancel if still running:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TransformationJobStatus
>>> c = CogniteClient()
>>>
>>> res = c.transformations.run(id = 1, timeout = 60.0)
>>> if res.status == TransformationJobStatus.RUNNING:
>>>     res.cancel()
```

List transformations

```

TransformationsAPI.list (include_public: bool = True, name_regex: Optional[str] =
None, query_regex: Optional[str] = None, destination_type:
Optional[str] = None, conflict_mode: Optional[str] = None,
cdf_project_name: Optional[str] = None, has_blocked_error:
Optional[bool] = None, created_time: Union[Dict[str,
Any],
cognite.client.data_classes.shared.TimestampRange,
None] = None, last_updated_time: Union[Dict[str,
Any],
cognite.client.data_classes.shared.TimestampRange,
None]
= None, data_set_ids: Optional[List[int]] = None,
data_set_external_ids: Optional[List[str]] = None, tags: Op-
tional[cognite.client.data_classes.transformations.TagsFilter]
= None, limit: Optional[int] = 25) → cog-
nite.client.data_classes.transformations.TransformationList

```

List all transformations.

Parameters

- **include_public** (*bool*) – Whether public transformations should be included in the results. (default true).
- **name_regex** (*str*) – Regex expression to match the transformation name
- **query_regex** (*str*) – Regex expression to match the transformation query
- **destination_type** (*str*) – Transformation destination resource name to filter by.
- **conflict_mode** (*str*) – Filters by a selected transformation action type: abort/create, upsert, update, delete
- **cdf_project_name** (*str*) – Project name to filter by configured source and destination project
- **has_blocked_error** (*bool*) – Whether only the blocked transformations should be included in the results.
- **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[int]*) – Return only transformations in the specified data sets with these ids.
- **data_set_external_ids** (*List[str]*) – Return only transformations in the specified data sets with these external ids.
- **tags** (*TagsFilter*) – Return only the resource matching the specified tags constraints. It only supports ContainsAny as of now.
- **limit** (*int*) – Limits the number of results to be returned. To retrieve all results use limit=-1, default limit is 25.

Returns List of transformations

Return type *TransformationList*

Example

List transformations:

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

Update transformations

TransformationsAPI.**update** (*item: Union[cognite.client.data_classes.transformations.Transformation, cognite.client.data_classes.transformations.TransformationUpdate, Sequence[Union[cognite.client.data_classes.transformations.Transformation, cognite.client.data_classes.transformations.TransformationUpdate]]]*)
→ Union[cognite.client.data_classes.transformations.Transformation, cognite.client.data_classes.transformations.TransformationList]

Update one or more transformations

Parameters *item* (Union[Transformation, TransformationUpdate, List[Union[Transformation, TransformationUpdate]]]) – Transformation(s) to update

Returns Updated transformation(s)

Return type Union[Transformation, TransformationList]

Examples

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

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> transformation = c.transformations.retrieve(id=1)
>>> transformation.query = "SELECT * FROM _cdf.assets"
>>> res = c.transformations.update(transformation)
```

Perform a partial update on a transformation, updating the query and making it private:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TransformationUpdate
>>> c = CogniteClient()
>>> my_update = TransformationUpdate(id=1).query.set("SELECT * FROM _cdf.assets").
→ is_public.set(False)
>>> res = c.transformations.update(my_update)
```

Delete transformations

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

Delete one or more transformations.

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

Example

Delete transformations by id or external id:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> c.transformations.delete(id=[1,2,3], external_id="function3")
```

Transformation Schedules

Create transformation Schedules

TransformationSchedulesAPI.**create** (*schedule: Union[cognite.client.data_classes.transformations.schedules.TransformationSchedule, Sequence[cognite.client.data_classes.transformations.schedules.TransformationSchedule]]* → *Union[cognite.client.data_classes.transformations.schedules.TransformationSchedule, cognite.client.data_classes.transformations.schedules.TransformationScheduleList]*)

Schedule the specified transformation with the specified configuration(s).

Parameters **schedule** (*Union[TransformationSchedule, Sequence[TransformationSchedule]]*) – Configuration or list of configurations of the schedules to create.

Returns Created schedule(s)

Examples

Create new schedules:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TransformationSchedule
>>> c = CogniteClient()
>>> schedules = [TransformationSchedule(id = 1, interval = "0 * * * *"),
↳ TransformationSchedule(external_id="transformation2", interval = "5 * * * *")]
>>> res = c.transformations.schedules.create(schedules)
```

Retrieve transformation schedules

TransformationSchedulesAPI.**retrieve** (*id: Optional[int] = None, external_id: Optional[str] = None*) → *Optional[cognite.client.data_classes.transformations.schedules.TransformationSchedule]*

Retrieve a single transformation schedule by the id or external id of its transformation.

Parameters

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

Returns Requested transformation schedule or None if it does not exist.

Return type Optional[*TransformationSchedule*]

Examples

Get transformation schedule by transformation id:

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

Get transformation schedule by transformation external id:

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

Retrieve multiple transformation schedules

`TransformationSchedulesAPI.retrieve_multiple` (*ids*: Optional[Sequence[int]] = None, *external_ids*: Optional[Sequence[str]] = None, *ignore_unknown_ids*: bool = False) → `cognite.client.data_classes.transformations.schedules.TransformationScheduleList`

Retrieve multiple transformation schedules by the ids or external ids of the corresponding transformations.

Parameters

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

Returns Requested transformation schedules.

Return type *TransformationScheduleList*

Examples

Get transformation schedules by transformation ids:

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

Get transformation schedules by transformation external ids:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> res = c.transformations.schedules.retrieve_multiple(external_ids=["t1", "t2"])
```

List transformation schedules

`TransformationSchedulesAPI.list` (*include_public*: `bool` = `True`,
limit: `Optional[int]` = `25`) → `cognite.client.data_classes.transformations.schedules.TransformationScheduleList`

List all transformation schedules.

Parameters

- **include_public** (`bool`) – Whether public transformations should be included in the results. (default true).
- **cursor** (`str`) – Cursor for paging through results.
- **limit** (`int`) – Limits the number of results to be returned. To retrieve all results use `limit=-1`, default limit is 25.

Returns List of schedules

Return type `TransformationScheduleList`

Example

List schedules:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> schedules_list = c.transformations.schedules.list()
```

Update transformation schedules

`TransformationSchedulesAPI.update` (*item*: `Union[cognite.client.data_classes.transformations.schedules.TransformationSchedule, cognite.client.data_classes.transformations.schedules.TransformationScheduleUpdate, Sequence[Union[cognite.client.data_classes.transformations.schedules.TransformationSchedule, cognite.client.data_classes.transformations.schedules.TransformationScheduleUpdate]]]` → `Union[cognite.client.data_classes.transformations.schedules.TransformationSchedule, cognite.client.data_classes.transformations.schedules.TransformationScheduleList]`)

Update one or more transformation schedules

Parameters *item* (`Union[TransformationSchedule, TransformationScheduleUpdate, Sequence[Union[TransformationSchedule, TransformationScheduleUpdate]]]`)
 – Transformation schedule(s) to update

Returns Updated transformation schedule(s)

Return type `Union[TransformationSchedule, TransformationScheduleList]`

Examples

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

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> transformation_schedule = c.transformations.schedules.retrieve(id=1)
>>> transformation_schedule.is_paused = True
>>> res = c.transformations.schedules.update(transformation_schedule)
```

Perform a partial update on a transformation schedule, updating the interval and unpausing it:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TransformationScheduleUpdate
>>> c = CogniteClient()
>>> my_update = TransformationScheduleUpdate(id=1).interval.set("0 * * * *").is_
↳ paused.set(False)
>>> res = c.transformations.schedules.update(my_update)
```

Delete transformation schedules

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

Unschedule one or more transformations

Parameters

- **id** (*Union[int, Sequence[int]]*) – Id or list of ids
- **external_id** (*Union[str, Sequence[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 schedules by id or external id:

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

Transformation Notifications

Create transformation notifications

`TransformationNotificationsAPI.create` (*notification: Union[cognite.client.data_classes.transformations.notifications.TransformationNotification, Sequence[cognite.client.data_classes.transformations.notifications.TransformationNotification]]*) → *Union[cognite.client.data_classes.transformations.notifications.TransformationNotification, Sequence[cognite.client.data_classes.transformations.notifications.TransformationNotification]]*

Subscribe for notifications on the transformation errors.

Parameters **notification** (*Union[TransformationNotification, Sequence[TransformationNotification]]*) – Notification or list of notifications to create.

Returns Created notification(s)

Examples

Create new notifications:

```

>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TransformationNotification
>>> c = CogniteClient()
>>> notifications = [TransformationNotification(transformation_id = 1,
↳ destination="my@email.com"), TransformationNotification(transformation_external_
↳ id="transformation2", destination="other@email.com")]
>>> res = c.transformations.notifications.create(notifications)

```

List transformation notifications

TransformationNotificationsAPI.**list** (*transformation_id: Optional[int] = None, transformation_external_id: Optional[str] = None, destination: Optional[str] = None, limit: Optional[int] = 25*) → cognite.client.data_classes.transformations.notifications.TransformationNotificationList

List notification subscriptions.

Parameters

- **transformation_id** (*Optional[int]*) – Filter by transformation internal numeric ID.
- **transformation_external_id** (*str*) – Filter by transformation externalId.
- **destination** (*str*) – Filter by notification destination.
- **limit** (*int*) – Limits the number of results to be returned. To retrieve all results use `limit=-1`, default limit is 25.

Returns List of transformation notifications

Return type *TransformationNotificationList*

Example

List all notifications:

```

>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> notifications_list = c.transformations.notifications.list()

```

List all notifications by transformation id:

```

>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> notifications_list = c.transformations.notifications.list(transformation_id =
↳ 1)

```

Delete transformation notifications

TransformationNotificationsAPI.**delete** (*id: Union[int, Sequence[int], None] = None*) → None
 Deletes the specified notification subscriptions on the transformation. Does nothing when the subscriptions already don't exist

Parameters **id** (*Union[int, Sequence[int]*) – Id or list of transformation notification ids

Returns None

Examples

Delete schedules by id or external id:

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

Transformation Jobs

Retrieve transformation jobs

`TransformationJobsAPI.retrieve` (*id: int*) → `Optional[cognite.client.data_classes.transformations.jobs.TransformationJob]`
Retrieve a single transformation job by id.

Parameters `id` (*int*) – Job internal Id

Returns Requested transformation job or None if it does not exist.

Return type `Optional[TransformationJob]`

Examples

Get transformation job by id:

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

`TransformationJobsAPI.retrieve_multiple` (*ids: Sequence[int], ignore_unknown_ids: bool = False*) → `cognite.client.data_classes.transformations.jobs.TransformationJobList`

Retrieve multiple transformation jobs by id.

Parameters

- **ids** (*Sequence[int]*) – Job internal Ids
- **ignore_unknown_ids** (*bool*) – Ignore IDs that are not found rather than throw an exception.

Returns Requested transformation jobs.

Return type `TransformationJobList`

Examples

Get jobs by id:

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

List transformation jobs

`TransformationJobsAPI.list` (*limit*: `Optional[int] = 25`, *transformation_id*: `Optional[int] = None`, *transformation_external_id*: `Optional[str] = None`) → `cognite.client.data_classes.transformations.jobs.TransformationJobList`

List all running transformation jobs.

Parameters

- **limit** (*int*) – Limits the number of results to be returned. To retrieve all results use `limit=-1`, default limit is 25.
- **transformation_id** (*int*) – Filters the results by the internal transformation id.
- **transformation_external_id** (*str*) – Filters the results by the external transformation id.

Returns List of transformation jobs

Return type `TransformationJobList`

Example

List transformation jobs:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> transformation_jobs_list = c.transformations.jobs.list()
```

List transformation jobs of a single transformation:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>> transformation_jobs_list = c.transformations.jobs.list(transformation_id = 1)
```

Transformation Schema

Get transformation schema

`TransformationSchemaAPI.retrieve` (*destination*: `cognite.client.data_classes.transformations.common.TransformationDestination`, *conflict_mode*: `Optional[str] = None`) → `cognite.client.data_classes.transformations.schema.TransformationSchemaColumnList`

Get expected schema for a transformation destination.

Parameters

- **destination** (`TransformationDestination`) – destination for which the schema is requested.
- **conflict_mode** (`Optional[str]`) – conflict mode for which the schema is requested.

Returns List of column descriptions

Return type `TransformationSchemaColumnList`

Example

Get the schema for a transformation producing assets:

```
>>> from cognite.client import CogniteClient
>>> from cognite.client.data_classes import TransformationDestination
>>> c = CogniteClient()
>>> columns = c.transformations.schema.retrieve(destination = TransformationDestination.assets())
```

Data classes

```
class cognite.client.data_classes.transformations.ContainsAny (tags: Optional[List[str]] = None)
```

Bases: `cognite.client.data_classes.transformations.TagsFilter`

Return transformations that has one of the tags specified.

Parameters `tags` (`List[str]`) – The resource item contains at least one of the listed tags. The tags are defined by a list of external ids.

Examples

List only resources marked as a PUMP or as a VALVE:

```
>>> from cognite.client.data_classes import ContainsAny
>>> my_tag_filter = ContainsAny(tags=["PUMP", "VALVE"])
```

```
class cognite.client.data_classes.transformations.SessionDetails (session_id: Optional[int] = None, client_id: Optional[str] = None, project_name: Optional[str] = None)
```

Bases: `object`

Details of a session which provid.

Parameters

- **session_id** (`int`) – CDF session ID
- **client_id** (`str`) – Idp client ID
- **project_name** (`str`) – CDF project name

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]`

```

class cognite.client.data_classes.transformations.Transformation (id: int =
    None, external_id:
    str = None, name: str =
    None, query:
    str = None, destination:
    TransformationDestination =
    None, conflict_mode:
    str = None, is_public:
    bool =
    True, ignore_null_fields:
    bool = False, source_api_key:
    str = None, destina-
    tion_api_key:
    str = None, source_oidc_credentials:
    Optional[OidcCredentials]
    = None, destina-
    tion_oidc_credentials:
    Optional[OidcCredentials]
    = None, created_time:
    Optional[int]
    = None, last_updated_time:
    Optional[int]
    = None, owner:
    str = None, owner_is_current_user:
    bool = True, has_source_api_key:
    Optional[bool]
    = None, has_destination_api_key:
    Optional[bool]
    = None, has_source_oidc_credentials:
    Optional[bool]
    = None, has_destination_oidc_credentials:
    Optional[bool]
    = None, running_job:

```

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

The transformations resource allows transforming data in CDF.

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.
- **name** (*str*) – The name of the Transformation.
- **query** (*str*) – SQL query of the transformation.
- **destination** (`TransformationDestination`) – see `TransformationDestination` for options.
- **conflict_mode** (*str*) – What to do in case of id collisions: either “abort”, “upsert”, “update” or “delete”
- **is_public** (*bool*) – Indicates if the transformation is visible to all in project or only to the owner.
- **ignore_null_fields** (*bool*) – Indicates how null values are handled on updates: ignore or set null.
- **source_api_key** (*str*) – Configures the transformation to authenticate with the given api key on the source.
- **destination_api_key** (*str*) – Configures the transformation to authenticate with the given api key on the destination.
- **source_oidc_credentials** (`Optional[OidcCredentials]`) – Configures the transformation to authenticate with the given oidc credentials key on the destination.
- **destination_oidc_credentials** (`Optional[OidcCredentials]`) – Configures the transformation to authenticate with the given oidc credentials on the destination.
- **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.
- **owner** (*str*) – Owner of the transformation: requester’s identity.
- **owner_is_current_user** (*bool*) – Indicates if the transformation belongs to the current user.
- **has_source_api_key** (*bool*) – Indicates if the transformation is configured with a source api key.
- **has_destination_api_key** (*bool*) – Indicates if the transformation is configured with a destination api key.
- **has_source_oidc_credentials** (*bool*) – Indicates if the transformation is configured with a source oidc credentials set.
- **has_destination_oidc_credentials** (*bool*) – Indicates if the transformation is configured with a destination oidc credentials set.
- **running_job** (`TransformationJob`) – Details for the job of this transformation currently running.

- **last_finished_job** (`TransformationJob`) – Details for the last finished job of this transformation.
- **blocked** (`TransformationBlockedInfo`) – Provides reason and time if the transformation is blocked.
- **schedule** (`TransformationSchedule`) – Details for the schedule if the transformation is scheduled.
- **cognite_client** (`CogniteClient`) – The client to associate with this object.
- **source_nonce** (`NonceCredentials`) – Single use credentials to bind to a CDF session for reading.
- **destination_nonce** (`NonceCredentials`) – Single use credentials to bind to a CDF session for writing.
- **source_session** (`SessionDetails`) – Details for the session used to read from the source project.
- **destination_session** (`SessionDetails`) – Details for the session used to write to the destination project.

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]

```

class cognite.client.data_classes.transformations.TransformationFilter (include_public:
    bool
    =
    True,
    name_regex:
    Op-
    tional[str]
    =
    None,
    query_regex:
    Op-
    tional[str]
    =
    None,
    des-
    ti-
    na-
    tion_type:
    Op-
    tional[str]
    =
    None,
    con-
    flict_mode:
    Op-
    tional[str]
    =
    None,
    cdf_project_name:
    Op-
    tional[str]
    =
    None,
    has_blocked_error:
    Op-
    tional[bool]
    =
    None,
    cre-
    ated_time:
    Union[Dict[str,
    Any],
    cog-
    nite.client.data_classes.share
    None]
    =
    None,
    last_updated_time:
    Union[Dict[str,
    Any],
    cog-
    nite.client.data_classes.share
    None]
    =
    None,
    data_set_ids:
    Op-
    tional[List[Dict[str,
    Any]]]
    =

```

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

No description.

Parameters

- **include_public** (*bool*) – Whether public transformations should be included in the results. The default is true.
- **name_regex** (*str*) – Regex expression to match the transformation name
- **query_regex** (*str*) – Regex expression to match the transformation query
- **destination_type** (*str*) – Transformation destination resource name to filter by.
- **conflict_mode** (*str*) – Filters by a selected transformation action type: abort/create, upsert, update, delete
- **cdf_project_name** (*str*) – Project name to filter by configured source and destination project
- **has_blocked_error** (*bool*) – Whether only the blocked transformations should be included in the results.
- **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]]*) – Return only transformations in the specified data sets with these ids.
- **tags** (*TagsFilter*) – Return only the resource matching the specified tags constraints. It only supports ContainsAny as of now.

dump (*camel_case: bool = True*) → `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]`

```
class cognite.client.data_classes.transformations.TransformationList (resources:
    Collection[Any],
    cog-
    nite_client:
    Cog-
    nite-
    Client
    =
    None)
```

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

```
class cognite.client.data_classes.transformations.TransformationPreviewResult (schema:  
                                                                    Trans-  
                                                                    for-  
                                                                    ma-  
                                                                    tion-  
                                                                    Schema-  
                                                                    Column-  
                                                                    List  
                                                                    =  
                                                                    None,  
                                                                    re-  
                                                                    sults:  
                                                                    List[Dict]  
                                                                    =  
                                                                    None,  
                                                                    cog-  
                                                                    nite_client:  
                                                                    Cog-  
                                                                    nite-  
                                                                    Client  
                                                                    =  
                                                                    None)
```

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

Allows previewing the result of a sql transformation before executing it.

Parameters

- **schema** (`TransformationSchemaColumnList`) – List of column descriptions.
- **results** (`List[Dict]`) – List of resulting rows. Each row is a dictionary where the key is the column name and the value is the entrie.

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]`

```
class cognite.client.data_classes.transformations.TransformationUpdate (id:  
                                                                    Op-  
                                                                    tional[int]  
                                                                    =  
                                                                    None,  
                                                                    ex-  
                                                                    ter-  
                                                                    nal_id:  
                                                                    Op-  
                                                                    tional[str]  
                                                                    =  
                                                                    None)
```

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

Changes applied to transformation

Parameters

- **id** (*int*) – A server-generated ID for the object.
- **external_id** (*str*) – External Id provided by client. Should be unique within the project.

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

Dump the instance into a json serializable Python data type.

Returns A dictionary representation of the instance.

Return type Dict[str, Any]

```
class cognite.client.data_classes.transformations.schedules.TransformationSchedule (id:
    int
    =
    None,
    external_id:
    str
    =
    None,
    created_time:
    int
    =
    None,
    last_updated:
    int
    =
    None,
    interval:
    str
    =
    None,
    is_paused:
    bool
    =
    False,
    cognite_client:
    CogniteClient
    =
    None)
```

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

The transformation schedules resource allows running recurrent transformations.

Parameters

- **id** (*int*) – Transformation id.
- **external_id** (*str*) – Transformation externalId.
- **created_time** (*int*) – Time when the schedule was created.

- **last_updated_time** (*int*) – Time when the schedule was last updated.
- **interval** (*str*) – Cron expression describes when the function should be called. Use <http://www.cronmaker.com> to create a cron expression.
- **is_paused** (*bool*) – If true, the transformation is not scheduled.
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

```
class cognite.client.data_classes.transformations.schedules.TransformationScheduleList (resource_collection[Transformation])
    = None
```

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

```
class cognite.client.data_classes.transformations.schedules.TransformationScheduleUpdate (id, operation, external_id)
    = None
```

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

Changes applied to transformation schedule

Parameters

- **id** (*int*) – Transformation id.
- **external_id** (*str*) – Transformation externalId.

```
class cognite.client.data_classes.transformations.notifications.TransformationNotification
```

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

The transformation notification resource allows configuring email alerts on events related to a transformation run.

Parameters

- **id** (*int*) – A server-generated ID for the object.
- **transformation_id** (*int*) – Transformation Id.
- **transformation_external_id** (*str*) – Transformation external Id.
- **destination** (*str*) – Email address where notifications should be sent.
- **created_time** (*int*) – Time when the notification was created.

- **last_updated_time** (*int*) – Time when the notification was last updated.
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

class `cognite.client.data_classes.transformations.notifications.TransformationNotification`

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

Parameters

- **transformation_id** (*Optional[int]*) – Filter by transformation internal numeric ID.
- **transformation_external_id** (*str*) – Filter by transformation externalId.
- **destination** (*str*) – Filter by notification destination.

class `cognite.client.data_classes.transformations.notifications.TransformationNotification`

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

```

class cognite.client.data_classes.transformations.jobs.TransformationJob (id:
    int
    =
    None,
    status:
    TransformationJobStatus
    =
    None,
    transformation_id:
    int
    =
    None,
    transformation_external_id:
    str
    =
    None,
    source_project:
    str
    =
    None,
    destination_project:
    str
    =
    None,
    destination:
    TransformationDestination
    =
    None,
    conflict_mode:
    str 279
    =
    None,
    query:

```

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

The transformation job resource allows following the status of execution of a transformation run.

Parameters

- **id** (*int*) – A server-generated ID for the object.
- **status** (`TransformationJobStatus`) – Status of the job.
- **transformation_id** (*int*) – Server-generated ID of the transformation.
- **transformation_external_id** (*str*) – external ID of the transformation.
- **source_project** (*str*) – Name of the CDF project the data will be read from.
- **destination_project** (*str*) – Name of the CDF project the data will be written to.
- **destination_type** (*str*) – Target resource type of the transformation.
- **destination_database** (*str*) – Target database if the destination type is raw.
- **destination_table** (*str*) – Target table name if the destination type is RAW.
- **conflict_mode** (*str*) – What to do in case of id collisions: either “abort”, “upsert”, “update” or “delete”.
- **query** (*str*) – Query of the transformation that is being executed.
- **error** (*str*) – Error message from the server.
- **ignore_null_fields** (*bool*) – Indicates how null values are handled on updates: ignore or set null.
- **created_time** (*int*) – Time when the job was created.
- **started_time** (*int*) – Time when the job started running.
- **finished_time** (*int*) – Time when the job finished running.
- **last_seen_time** (*int*) – Time of the last status update from the job.
- **cognite_client** (`CogniteClient`) – The client to associate with this object.

metrics () → `cognite.client.data_classes.transformations.jobs.TransformationJobMetricList`
Get job metrics.

update () → `None`
Get updated job status.

wait (*polling_interval*: *float* = 1, *timeout*: *Optional[float]* = None) → `cognite.client.data_classes.transformations.jobs.TransformationJob`
Waits for the job to finish.

Parameters

- **polling_interval** (*float*) – time (s) to wait between job status updates, default is one second.
- **timeout** (*Optional[float]*) – maximum time (s) to wait, default is None (infinite time). Once the timeout is reached, it returns with the current status.

Returns self.

Return type `TransformationJob`

Examples

run transformations 1 and 2 in parallel, and run 3 once they finish successfully:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>>
>>> job1 = c.transformations.run(id = 1, wait = False)
>>> job2 = c.transformations.run(id = 2, wait = False)
>>> job1.wait()
>>> job2.wait()
>>> if TransformationJobStatus.FAILED not in [job1.status, job2.status]:
>>>     c.transformations.run(id = 3, wait = False)
```

wait transformation for 5 minutes and do something if still running:

```
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>>
>>> job = c.transformations.run(id = 1, wait = False)
>>> job.wait(timeout = 5.0*60)
>>> if job.status == TransformationJobStatus.FAILED:
>>>     # do something if job failed
>>> elif job.status == TransformationJobStatus.COMPLETED:
>>>     # do something if job completed successfully
>>> else:
>>>     # do something if job is still running
```

`wait_async` (*polling_interval*: float = 1, *timeout*: Optional[float] = None) → `cognite.client.data_classes.transformations.jobs.TransformationJob`
 Asyncio coroutine, waits for the job to finish asynchronously.

Parameters

- **polling_interval** (float) – time (s) to wait between job status updates, default is one second.
- **timeout** (Optional[float]) – maximum time (s) to wait, default is None (infinite time). Once the timeout is reached, it returns with the current status.

Returns coroutine object that will finish when the job finishes and resolves to self.

Return type Awaitable[*TransformationJob*]

Examples

run transformations 1 and 2 in parallel, and run 3 once they finish successfully:

```
>>> from asyncio import ensure_future
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>>
>>> async def run_successive_transformations():
>>>     job1 = c.transformations.run(id = 1, wait = False)
>>>     job2 = c.transformations.run(id = 2, wait = False)
>>>     await job1.wait_async()
>>>     await job2.wait_async()
>>>     if TransformationJobStatus.FAILED not in [job1.status, job2.status]:
```

(continues on next page)

(continued from previous page)

```
>>> c.transformations.run(id = 3, wait = False)
>>>
>>> ensure_future(run_successive_transformations())
```

wait transformation for 5 minutes and do something if still running:

```
>>> from asyncio import ensure_future
>>> from cognite.client import CogniteClient
>>> c = CogniteClient()
>>>
>>> async def run_successive_transformations():
>>>     job = c.transformations.run(id = 1, wait = False)
>>>     await job.wait_async(timeout = 5.0*60)
>>>     if job.status == TransformationJobStatus.FAILED:
>>>         # do something if job failed
>>>     elif job.status == TransformationJobStatus.COMPLETED:
>>>         # do something if job completed successfully
>>>     else:
>>>         # do something if job is still running
>>>
>>> ensure_future(run_successive_transformations())
```

class `cognite.client.data_classes.transformations.jobs.TransformationJobFilter` (*transformation_id: Optional[int]*, *transformation_external_id: Optional[str]*) = *None*

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

Parameters

- **transformation_id** (*Optional[int]*) – Filter jobs by transformation internal numeric ID.
- **transformation_external_id** (*str*) – Filter jobs by transformation external ID.

class `cognite.client.data_classes.transformations.jobs.TransformationJobList` (*resources: Collection[Any]*, *cognite_client: CogniteClient*) = *None*

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

```

class cognite.client.data_classes.transformations.jobs.TransformationJobMetric (id:
                                                                    int
                                                                    =
                                                                    None,
                                                                    times-
                                                                    tamp:
                                                                    int
                                                                    =
                                                                    None,
                                                                    name:
                                                                    str
                                                                    =
                                                                    None,
                                                                    count:
                                                                    int
                                                                    =
                                                                    None,
                                                                    cog-
                                                                    nite_client:
                                                                    Cog-
                                                                    nite-
                                                                    Client
                                                                    =
                                                                    None)

```

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

The transformation job metric resource allows following details of execution of a transformation run.

Parameters

- **timestamp** (*int*) – Time of the last metric update.
- **name** (*str*) – Name of the metric.
- **count** (*int*) – Value of the metric.
- **cognite_client** (`CogniteClient`) – The client to associate with this object.

```

class cognite.client.data_classes.transformations.jobs.TransformationJobMetricList (resources:
                                                                                          Col-
                                                                                          lec-
                                                                                          tion[Any],
                                                                                          cog-
                                                                                          nite_client:
                                                                                          Cog-
                                                                                          nite-
                                                                                          Client
                                                                                          =
                                                                                          None)

```

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

```

class cognite.client.data_classes.transformations.jobs.TransformationJobStatus
Bases: str, enum.Enum

```

An enumeration.

```
class cognite.client.data_classes.transformations.schema.TransformationSchemaColumn (name:
    str
    =
    None,
    sql_type:
    str
    =
    None,
    type:
    Trans-
    for-
    ma-
    tion-
    SchemaTy
    =
    None,
    nul-
    lable:
    bool
    =
    False,
    cog-
    nite_client:
    Cog-
    nite-
    Client
    =
    None)
```

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

Represents a column of the expected sql structure for a destination type.

Parameters

- **name** (*str*) – Column name
- **sql_type** (*str*) – Type of the column in sql format.
- **type** (*TransformationSchemaType*) – Type of the column in json format.
- **nullable** (*bool*) – Values for the column can be null or not
- **cognite_client** (*CogniteClient*) – The client to associate with this object.

```
class cognite.client.data_classes.transformations.schema.TransformationSchemaColumnList (resou
    Col
    lec-
    tion
    cog
    nite
    Cog
    nite
    Cli
    =
    Non
```

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

```

class cognite.client.data_classes.transformations.common.DataModelInstances (model_external_id:
    Optional[str]
    =
    None,
    space_external_id:
    Optional[str]
    =
    None,
    instance_space_external_id:
    Optional[str]
    =
    None)
    Bases:
        cognite.client.data_classes.transformations.common.
        TransformationDestination

```

```

class cognite.client.data_classes.transformations.common.InstanceEdges (view:
    Optional[cognite.client.data_classes.transformations.common.InstanceEdgesView]
    =
    None,
    instance_space:
    Optional[str]
    =
    None,
    edge_type:
    Optional[cognite.client.data_classes.transformations.common.InstanceEdgeType]
    =
    None)
    Bases:
        cognite.client.data_classes.transformations.common.
        TransformationDestination

```

```

class cognite.client.data_classes.transformations.common.InstanceNodes (view:
    Optional[cognite.client.data_classes.transformations.common.InstanceNodesView]
    =
    None,
    instance_space:
    Optional[str]
    =
    None)
    Bases:
        cognite.client.data_classes.transformations.common.
        TransformationDestination

```

```
class cognite.client.data_classes.transformations.common.RawTable (database:
                                                                    Op-
                                                                    tional[str]
                                                                    = None,
                                                                    table: Op-
                                                                    tional[str]
                                                                    = None)
Bases: cognite.client.data_classes.transformations.common.
TransformationDestination
```

```
class cognite.client.data_classes.transformations.common.SequenceRows (external_id:
                                                                    Op-
                                                                    tional[str]
                                                                    =
                                                                    None)
Bases: cognite.client.data_classes.transformations.common.
TransformationDestination
```

```
class cognite.client.data_classes.transformations.common.TransformationBlockedInfo (reason:
                                                                    Op-
                                                                    tional[str]
                                                                    =
                                                                    None,
                                                                    cre-
                                                                    ated_time:
                                                                    Op-
                                                                    tional[int]
                                                                    =
                                                                    None)
Bases: object
```

Information about the reason why and when a transformation is blocked.

Parameters

- **reason** (*str*) – Reason why the transformation is blocked.
- **created_time** (*Optional[int]*) – Timestamp when the transformation was blocked.

```
class cognite.client.data_classes.transformations.common.TransformationDestination (type:
                                                                    Op-
                                                                    tional[str]
                                                                    =
                                                                    None)
Bases: object
```

TransformationDestination has static methods to define the target resource type of a transformation

Parameters **type** (*str*) – Used as data type identifier on transformation creation/retrieval.

static asset_hierarchy () → *cognite.client.data_classes.transformations.common.TransformationDestination*
 To be used when the transformation is meant to produce asset hierarchies.

static assets () → *cognite.client.data_classes.transformations.common.TransformationDestination*
 To be used when the transformation is meant to produce assets.

static data_model_instances (*model_external_id: str = "", space_external_id: str = "", instance_space_external_id: str = ""*) → *cognite.client.data_classes.transformations.common.DataModelInstances*

To be used when the transformation is meant to produce data model instances. Flexible Data Models resource type is on *beta* version currently.

Parameters

- **model_external_id** (*str*) – external_id of the flexible data model.
- **space_external_id** (*str*) – space external_id of the flexible data model.
- **instance_space_external_id** (*str*) – space external_id of the flexible data model instance.

Returns TransformationDestination pointing to the target flexible data model.

static data_sets () → cognite.client.data_classes.transformations.common.TransformationDestination
To be used when the transformation is meant to produce data sets.

static datapoints () → cognite.client.data_classes.transformations.common.TransformationDestination
To be used when the transformation is meant to produce numeric data points.

static events () → cognite.client.data_classes.transformations.common.TransformationDestination
To be used when the transformation is meant to produce events.

static files () → cognite.client.data_classes.transformations.common.TransformationDestination
To be used when the transformation is meant to produce files.

static instance_edges (*view: Optional[cognite.client.data_classes.transformations.common.ViewInfo] = None, instance_space: Optional[str] = None, edge_type: Optional[cognite.client.data_classes.transformations.common.EdgeType] = None*) → cognite.client.data_classes.transformations.common.InstanceEdges

To be used when the transformation is meant to produce edge’s instances. Flexible Data Models resource type is on *beta* version currently.

Parameters

- **view** (*ViewInfo*) – information of the view.
- **instance_space** (*str*) – space id of the instance.
- **edge_type** (*EdgeType*) – information about the type of the edge

Returns pointing to the target flexible data model.

Return type *InstanceEdges*

static instance_nodes (*view: Optional[cognite.client.data_classes.transformations.common.ViewInfo] = None, instance_space: Optional[str] = None*) → cognite.client.data_classes.transformations.common.InstanceNodes

To be used when the transformation is meant to produce node’s instances. Flexible Data Models resource type is on *beta* version currently.

Parameters

- **view** (*ViewInfo*) – information of the view.
- **instance_space** (*str*) – space id of the instance.

Returns pointing to the target flexible data model.

Return type *InstanceNodes*

static labels () → cognite.client.data_classes.transformations.common.TransformationDestination
To be used when the transformation is meant to produce labels.

static raw (*database: str = "", table: str = ""*) → `cognite.client.data_classes.transformations.common.RawTable`
To be used when the transformation is meant to produce raw table rows.

Parameters

- **database** (*str*) – database name of the target raw table.
- **table** (*str*) – name of the target raw table

Returns `TransformationDestination` pointing to the target table

static relationships () → `cognite.client.data_classes.transformations.common.TransformationDestination`
To be used when the transformation is meant to produce relationships.

static sequence_rows (*external_id: str = ""*) → `cognite.client.data_classes.transformations.common.SequenceRows`
To be used when the transformation is meant to produce sequence rows.

Parameters **external_id** (*str*) – Sequence external id.

Returns `TransformationDestination` pointing to the target sequence rows

static sequences () → `cognite.client.data_classes.transformations.common.TransformationDestination`
To be used when the transformation is meant to produce sequences.

static string_datapoints () → `cognite.client.data_classes.transformations.common.TransformationDestination`
To be used when the transformation is meant to produce string data points.

static timeseries () → `cognite.client.data_classes.transformations.common.TransformationDestination`
To be used when the transformation is meant to produce time series.

2.4.23 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: Sequence[str] = ('metadata',), ignore: List[str] = None, camel_case: bool = False*) → `pandas.DataFrame`
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: Collection[Any],
                                                           cognite_client: CogniteClient = 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]]

extend (*other: Collection[Any]*) → None

S.extend(iterable) – extend sequence by appending elements from the iterable

get (*id: Optional[int] = None, external_id: Optional[str] = None*) → Optional[cognite.client.data_classes._base.CogniteResource]

Get an item from this list by id or external_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: bool = False*) → 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*) → 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]

CogniteUpdate

```
class cognite.client.data_classes._base.CogniteUpdate (id: Optional[int] = None,  
external_id: Optional[str] =  
None)
```

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

Dump the instance into a json serializable Python data type.

Returns A dictionary representation of the instance.

Return type Dict[str, Any]

2.4.24 Exceptions

CogniteAPIError

```
exception cognite.client.exceptions.CogniteAPIError (message: str, code: int,  
x_request_id: Optional[str]  
= None, missing: Op-  
tional[Sequence[T_co]] =  
None, duplicated: Op-  
tional[Sequence[T_co]]  
= None, successful: Op-  
tional[Sequence[T_co]]  
= None, failed: Op-  
tional[Sequence[T_co]]  
= None, unknown: Op-  
tional[Sequence[T_co]]  
= None, unwrap_fn: Op-  
tional[Callable] = None, extra:  
Optional[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.
- **missing** – (List) List of missing identifiers.
- **duplicated** – (List) List of duplicated identifiers.
- **successful** (*List*) – List of items which were successfully processed.

- **failed** (*List*) – List of items which failed.
- **unknown** (*List*) – List of items which may or may not have been successfully processed.
- **unwrap_fn** – (Callable): Function to extract identifier from the Cognite resource.
- **extra** (*Dict*) – A dict of any additional information.

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()

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(f"Something went terribly wrong. Here is the request-id: {e.x_
→request_id}")
        print(f"The message returned from the API: {e.message}")
```

CogniteNotFoundError

exception `cognite.client.exceptions.CogniteNotFoundError` (*not_found*: *List*[*T*],
successful: *Optional*[*List*[*T*]] = *None*,
failed: *Optional*[*List*[*T*]] = *None*, *unknown*:
Optional[*List*[*T*]] = *None*, *unwrap_fn*: *Optional*[*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 processed.
- **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: `Optional[List[T]]` =
`None`, *failed*: `Optional[List[T]]` =
`None`, *unknown*:
`Optional[List[T]]` =
`None`, *unwrap_fn*:
`Optional[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 processed.
- **failed** (`List`) – List of items which failed.
- **unknown** (`List`) – List of items which may or may not have been successfully processed.
- **unwrap_fn** – (Callable): Function to extract identifier from the Cognite resource.

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*: `Optional[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.

2.4.25 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 valid Cognite timestamps, i.e. milliseconds since epoch, to datetime object.

Parameters `ms` (*Union[int, float]*) – Milliseconds since epoch.

Raises `ValueError` – On invalid Cognite timestamps.

Returns Aware datetime object in UTC.

Return type `datetime`

Convert datetime to milliseconds since epoch

`cognite.client.utils.datetime_to_ms` (*dt: datetime.datetime*) → `int`

Converts datetime object to milliseconds since epoch.

Parameters `dt` (*datetime*) – Naive or aware datetime object. Naive datetimes are interpreted as local time.

Returns Milliseconds since epoch (negative for time prior to 1970-01-01)

Return type `ms`

2.4.26 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` () → `Iterator[cognite.client.testing.CogniteClientMock]`

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
```

C

`cognite.client.data_classes.transformations.jobs`,
`cognite.client.data_classes.transformations.images`,
`cognite.client.data_classes.transformations.notifications`,
`cognite.client.data_classes.transformations.primitives`,
`cognite.client.data_classes.transformations.scheduled`,
`cognite.client.data_classes.transformations.schema`,
`cognite.client.data_classes.assets`,
`cognite.client.data_classes.contextualization`,
`cognite.client.data_classes.data_sets`,
`cognite.client.data_classes.datapoints`,
`cognite.client.data_classes.events`,
`cognite.client.data_classes.extractionpipelines`,
`cognite.client.data_classes.files`,
`cognite.client.data_classes.functions`,
`cognite.client.data_classes.geospatial`,
`cognite.client.data_classes.iam`,
`cognite.client.data_classes.labels`,
`cognite.client.data_classes.login`,
`cognite.client.data_classes.raw`,
`cognite.client.data_classes.relationships`,
`cognite.client.data_classes.sequences`,
`cognite.client.data_classes.templates`,
`cognite.client.data_classes.three_d`,
`cognite.client.data_classes.time_series`,
`cognite.client.data_classes.transformations`,
`cognite.client.data_classes.transformations.common`,

A

- `add_service_account()` (*cognite.client._api.iam.GroupsAPI* method), 234
- `aggregate()` (*cognite.client._api.assets.AssetsAPI* method), 18
- `aggregate()` (*cognite.client._api.data_sets.DataSetsAPI* method), 46
- `aggregate()` (*cognite.client._api.events.EventsAPI* method), 38
- `aggregate()` (*cognite.client._api.files.FilesAPI* method), 52
- `aggregate()` (*cognite.client._api.sequences.SequencesAPI* method), 112
- `aggregate()` (*cognite.client._api.time_series.TimeSeriesAPI* method), 81
- `aggregate_features()` (*cognite.client._api.geospatial.GeospatialAPI* method), 150
- `aggregate_metadata_keys()` (*cognite.client._api.assets.AssetsAPI* method), 19
- `aggregate_metadata_values()` (*cognite.client._api.assets.AssetsAPI* method), 19
- `aggregate_unique_values()` (*cognite.client._api.events.EventsAPI* method), 38
- `AggregateResultItem` (class in *cognite.client.data_classes.assets*), 26
- `Annotation` (class in *cognite.client.data_classes.annotations*), 227
- `AnnotationFilter` (class in *cognite.client.data_classes.annotations*), 228
- `AnnotationList` (class in *cognite.client.data_classes.annotations*), 229
- `AnnotationUpdate` (class in *cognite.client.data_classes.annotations*), 230
- `APIKey` (class in *cognite.client.data_classes.iam*), 237
- `APIKeyList` (class in *cognite.client.data_classes.iam*), 237
- `append()` (*cognite.client.data_classes.contextualization.Contextualization* method), 184
- `append()` (*cognite.client.data_classes.contextualization.DiagramConversion* method), 187
- `append()` (*cognite.client.data_classes.contextualization.EntityMatchingM* method), 194
- `Asset` (class in *cognite.client.data_classes.assets*), 26
- `asset()` (*cognite.client.data_classes.time_series.TimeSeries* method), 86
- `asset_hierarchy()` (*cognite.client.data_classes.transformations.common.Transformation* static method), 286
- `asset_subtree_ids` (*cognite.client.data_classes.contextualization.AssetTagDetectionParam* attribute), 181
- `ASSET_TAG_DETECTION` (*cognite.client.data_classes.contextualization.VisionFeature* attribute), 201
- `asset_tag_detection_parameters` (*cognite.client.data_classes.contextualization.FeatureParameters* attribute), 196
- `asset_tag_predictions` (*cognite.client.data_classes.contextualization.VisionExtractPrediction* attribute), 201
- `AssetAggregate` (class in *cognite.client.data_classes.assets*), 28
- `AssetFilter` (class in *cognite.client.data_classes.assets*), 28
- `AssetHierarchy` (class in *cognite.client.data_classes.assets*), 30
- `AssetLink` (class in *cognite.client.data_classes.annotation_types.images*), 203
- `AssetList` (class in *cognite.client.data_classes.assets*), 31
- `assets()` (*cognite.client.data_classes.transformations.common.Transform* static method), 286
- `AssetTagDetectionParameters` (class in *cog-*

nite.client.data_classes.contextualization), 181
 AssetUpdate (class in *cognite.client.data_classes.assets*), 31

B

beta_features (cognite.client.data_classes.contextualization.VisionFeature attribute), 202

bounding_box (cognite.client.data_classes.annotation_types.images.ObjectDetection attribute), 203

BoundingBox (class in *cognite.client.data_classes.annotation_types.primitives*), 204

BoundingBox3D (class in *cognite.client.data_classes.three_d*), 170

C

call () (*cognite.client._api.functions.FunctionsAPI* method), 68

call () (*cognite.client.data_classes.functions.Function* method), 75

cancel () (*cognite.client._api.transformations.TransformationsAPI* method), 258

CdfResourceRef (class in *cognite.client.data_classes.annotation_types.primitives*), 205

children () (*cognite.client.data_classes.assets.Asset* method), 27

clear () (*cognite.client.data_classes.contextualization.ContextualizationJobList* method), 184

clear () (*cognite.client.data_classes.contextualization.DiagramConvertPageList* method), 188

clear () (*cognite.client.data_classes.contextualization.EntityMatchingModelList* method), 194

ClientConfig (class in *cognite.client.config*), 10

ClientCredentials (class in *cognite.client.data_classes.iam*), 237

cognite.client.data_classes.annotation_types.images (module), 203

cognite.client.data_classes.annotation_types.primitives (module), 204

cognite.client.data_classes.annotations (module), 227

cognite.client.data_classes.assets (module), 26

cognite.client.data_classes.contextualization (module), 181

cognite.client.data_classes.data_sets (module), 47

cognite.client.data_classes.datapoints (module), 103

cognite.client.data_classes.events (module), 41

cognite.client.data_classes.extractionpipelines (module), 247

cognite.client.data_classes.files (module), 60

cognite.client.data_classes.functions (module), 74

cognite.client.data_classes.geospatial (module), 155

cognite.client.data_classes.iam (module), 237

cognite.client.data_classes.labels (module), 33

cognite.client.data_classes.login (module), 15

cognite.client.data_classes.raw (module), 129

cognite.client.data_classes.relationships (module), 136

cognite.client.data_classes.sequences (module), 119

cognite.client.data_classes.templates (module), 219

cognite.client.data_classes.three_d (module), 170

cognite.client.data_classes.time_series (module), 85

cognite.client.data_classes.transformations (module), 268

cognite.client.data_classes.transformations.common (module), 284

cognite.client.data_classes.transformations.jobs (module), 278

cognite.client.data_classes.transformations.notification (module), 276

cognite.client.data_classes.transformations.scheduled (module), 275

cognite.client.data_classes.transformations.schema (module), 283

CogniteAPIError, 290

CogniteAPIKeyError, 292

CogniteClient (class in *cognite.client*), 10

CogniteClientMock (class in *cognite.client.testing*), 293

CogniteDuplicatedError, 292

CogniteFilter (class in *cognite.client.data_classes._base*), 289

CogniteImportError, 292

CogniteMissingClientError, 292

CogniteNotFoundError, 291

CogniteResource (class in *cognite.client.data_classes._base*), 288

CogniteResourceList (class in *cognite.client.data_classes._base*), 289

CogniteResponse (class in *cognite*

`nite.client.data_classes._base`), 289

`CogniteUpdate` (class in `cognite.client.data_classes._base`), 290

`COLLECTING` (`cognite.client.data_classes.contextualization.JobStatus` attribute), 197

`column_external_ids` (`cognite.client.data_classes.sequences.SequenceData` attribute), 121

`column_value_types` (`cognite.client.data_classes.sequences.SequenceData` attribute), 121

`COMPLETED` (`cognite.client.data_classes.contextualization.JobStatus` attribute), 197

`compute()` (`cognite.client._api.geospatial.GeospatialAPI` method), 155

`concat_duplicate_ids()` (`cognite.client.data_classes.datapoints.DatapointsArrayList` method), 107

`confidence` (`cognite.client.data_classes.annotation_types.images.AssociatedImage` attribute), 203

`confidence` (`cognite.client.data_classes.annotation_types.images.TextRegion` attribute), 204

`config` (`cognite.client.CogniteClient` attribute), 10

`ConstantResolver` (class in `cognite.client.data_classes.templates`), 219

`ContainsAny` (class in `cognite.client.data_classes.transformations`), 268

`ContextualizationJob` (class in `cognite.client.data_classes.contextualization`), 182

`ContextualizationJobList` (class in `cognite.client.data_classes.contextualization`), 184

`ContextualizationJobType` (class in `cognite.client.data_classes.contextualization`), 185

`convert()` (`cognite.client._api.diagrams.DiagramsAPI` method), 179

`convert()` (`cognite.client.data_classes.contextualization.DiagramConversionPageList` method), 190

`CoordinateReferenceSystem` (class in `cognite.client.data_classes.geospatial`), 155

`CoordinateReferenceSystemList` (class in `cognite.client.data_classes.geospatial`), 155

`copy()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 184

`copy()` (`cognite.client.data_classes.contextualization.DiagramConversionPageList` method), 188

`copy()` (`cognite.client.data_classes.contextualization.EntityMatchingMethodList` method), 194

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 184

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 188

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 194

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 197

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 203

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 204

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 219

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 268

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 289

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 290

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 333

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 337

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 357

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 372

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 387

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 412

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 427

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 442

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 457

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 472

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 487

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 502

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 517

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 532

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 547

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 562

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 577

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 592

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 607

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 622

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 637

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 652

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 667

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 682

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 697

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 712

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 727

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 742

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 757

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 772

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 787

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 802

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 817

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 832

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 847

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 862

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 877

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 892

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 907

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 922

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 937

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 952

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 967

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 982

`count()` (`cognite.client.data_classes.contextualization.ContextualizationJobList` method), 997

`create()` (`cognite.client._api.annotations.AnnotationsAPI` method), 226

`create()` (`cognite.client._api.assets.AssetsAPI` method), 30

`create()` (`cognite.client._api.data_sets.DataSetsAPI` method), 46

`create()` (`cognite.client._api.events.EventsAPI` method), 39

`create()` (`cognite.client._api.extractionpipelines.ExtractionPipelineConfigurationList` method), 246

`create()` (`cognite.client._api.extractionpipelines.ExtractionPipelineRunsList` method), 244

`create()` (`cognite.client._api.extractionpipelines.ExtractionPipelinesAPI` method), 241

`create()` (`cognite.client._api.files.FilesAPI` method), 53

`create()` (`cognite.client._api.functions.FunctionsAPI` method), 64

`create()` (`cognite.client._api.functions.FunctionSchedulesAPI` method), 72

`create()` (`cognite.client._api.iam.APIKeysAPI` method), 232

`create()` (`cognite.client._api.iam.GroupsAPI` method), 233

`create()` (`cognite.client._api.iam.SecurityCategoriesAPI` method), 235

`create()` (`cognite.client._api.iam.ServiceAccountsAPI` method), 230

`create()` (`cognite.client._api.iam.SessionsAPI` method), 236

`create()` (`cognite.client._api.labels.LabelsAPI` method), 124

`create()` (`cognite.client._api.raw.RawDatabasesAPI` method), 124

`create()` (`cognite.client._api.raw.RawTablesAPI` method), 125

`create()` (`cognite.client._api.relationships.RelationshipsAPI` method), 133

`create()` (`cognite.client._api.sequences.SequencesAPI` method), 131

`create()` (`cognite.client._api.templates.TemplateGroupsAPI` method), 130

`create()` (`cognite.client._api.templates.TemplateInstancesAPI` method), 132

`create()` (`cognite.client._api.templates.TemplateViewsAPI` method), 136

[create\(\)](#) (*cognite.client._api.three_d.ThreeDAssetMappingAPI* [method](#)), 168
[create\(\)](#) (*cognite.client._api.three_d.ThreeDModelsAPI* [method](#)), 161
[create\(\)](#) (*cognite.client._api.three_d.ThreeDRevisionsAPI* [method](#)), 163
[create\(\)](#) (*cognite.client._api.time_series.TimeSeriesAPI* [method](#)), 83
[create\(\)](#) (*cognite.client._api.transformations.notifications.TransformationsNotificationsAPI* [method](#)), 264
[create\(\)](#) (*cognite.client._api.transformations.schedules.TransformationsSchedulesAPI* [method](#)), 261
[create\(\)](#) (*cognite.client._api.transformations.TransformationsAPI* [method](#)), 254
[create_coordinate_reference_systems\(\)](#) (*cognite.client._api.geospatial.GeospatialAPI* [method](#)), 151
[create_feature_types\(\)](#) (*cognite.client._api.geospatial.GeospatialAPI* [method](#)), 140
[create_features\(\)](#) (*cognite.client._api.geospatial.GeospatialAPI* [method](#)), 143
[create_hierarchy\(\)](#) (*cognite.client._api.assets.AssetsAPI* [method](#)), 22
[CreatedSession](#) (*class in cognite.client.data_classes.iam*), 237

D

[data_model_instances\(\)](#) (*cognite.client.data_classes.transformations.common.TransformationsCommonTransformationDestination* [static method](#)), 286
[data_sets\(\)](#) (*cognite.client.data_classes.transformations.common.TransformationsCommonTransformationDestination* [static method](#)), 287
[Database](#) (*class in cognite.client.data_classes.raw*), 129
[DatabaseList](#) (*class in cognite.client.data_classes.raw*), 130
[DataModelInstances](#) (*class in cognite.client.data_classes.transformations.common*), 284
[Datapoint](#) (*class in cognite.client.data_classes.datapoints*), 103
[Datapoints](#) (*class in cognite.client.data_classes.datapoints*), 104
[datapoints\(\)](#) (*cognite.client.data_classes.transformations.common.TransformationsCommonTransformationDestination* [static method](#)), 287
[DatapointsArray](#) (*class in cognite.client.data_classes.datapoints*), 106
[DatapointsArrayList](#) (*class in cognite.client.data_classes.datapoints*), 107
[default_client_config](#) (*cognite.client.config.GlobalConfig* [attribute](#)), 11
[delete\(\)](#) (*cognite.client._api.annotations.AnnotationsAPI* [method](#)), 227
[delete\(\)](#) (*cognite.client._api.assets.AssetsAPI* [method](#)), 24
[delete\(\)](#) (*cognite.client._api.entity_matching.EntityMatchingAPI* [method](#)), 176
[delete\(\)](#) (*cognite.client._api.events.EventsAPI* [method](#)), 40
[delete\(\)](#) (*cognite.client._api.extractionpipelines.ExtractionPipelinesAPI* [method](#)), 243
[delete\(\)](#) (*cognite.client._api.files.FilesAPI* [method](#)), 58
[delete\(\)](#) (*cognite.client._api.functions.FunctionsAPI* [method](#)), 66
[delete\(\)](#) (*cognite.client._api.functions.FunctionSchedulesAPI* [method](#)), 73
[delete\(\)](#) (*cognite.client._api.iam.APIKeysAPI* [method](#)), 232
[delete\(\)](#) (*cognite.client._api.iam.GroupsAPI* [method](#)), 233
[delete\(\)](#) (*cognite.client._api.iam.SecurityCategoriesAPI* [method](#)), 236
[delete\(\)](#) (*cognite.client._api.iam.ServiceAccountsAPI* [method](#)), 231
[delete\(\)](#) (*cognite.client._api.labels.LabelsAPI* [method](#)), 33
[delete\(\)](#) (*cognite.client._api.raw.RawDatabasesAPI* [method](#)), 124
[delete\(\)](#) (*cognite.client._api.raw.RawRowsAPI* [method](#)), 128
[delete\(\)](#) (*cognite.client._api.raw.RawTablesAPI* [method](#)), 125
[delete\(\)](#) (*cognite.client._api.relationships.RelationshipsAPI* [method](#)), 135
[delete\(\)](#) (*cognite.client._api.sequences.SequencesAPI* [method](#)), 113
[delete\(\)](#) (*cognite.client._api.sequences.SequencesDataAPI* [method](#)), 113

- method), 118
- delete() (cognite.client._api.templates.TemplateGroupsAPI method), 209
- delete() (cognite.client._api.templates.TemplateGroupVersionsAPI method), 211
- delete() (cognite.client._api.templates.TemplateInstancesAPI method), 216
- delete() (cognite.client._api.templates.TemplateViewsAPI method), 219
- delete() (cognite.client._api.three_d.ThreeDAssetMappingAPI method), 169
- delete() (cognite.client._api.three_d.ThreeDModelsAPI method), 162
- delete() (cognite.client._api.three_d.ThreeDRevisionsAPI method), 165
- delete() (cognite.client._api.time_series.TimeSeriesAPI method), 83
- delete() (cognite.client._api.transformations.notifications.TransformationsAPI method), 265
- delete() (cognite.client._api.transformations.schedules.TransformationsAPI method), 264
- delete() (cognite.client._api.transformations.TransformationsAPI method), 260
- delete() (cognite.client.CogniteClient method), 10
- delete_feature_types() (cognite.client._api.geospatial.GeospatialAPI method), 141
- delete_features() (cognite.client._api.geospatial.GeospatialAPI method), 143
- delete_range() (cognite.client._api.datapoints.DatapointsAPI method), 102
- delete_range() (cognite.client._api.sequences.SequencesDataAPI method), 119
- delete_ranges() (cognite.client._api.datapoints.DatapointsAPI method), 102
- delete_raster() (cognite.client._api.geospatial.GeospatialAPI method), 153
- description (cognite.client.data_classes.contextualization.EntityMetadata attribute), 195
- detect() (cognite.client._api.diagrams.DiagramsAPI method), 178
- DetectJobBundle (class in cognite.client.data_classes.contextualization), 185
- DiagramConvertItem (class in cognite.client.data_classes.contextualization), 186
- DiagramConvertPageList (class in cognite.client.data_classes.contextualization), 186
- DiagramConvertResults (class in cognite.client.data_classes.contextualization), 188
- DiagramDetectItem (class in cognite.client.data_classes.contextualization), 189
- DiagramDetectResults (class in cognite.client.data_classes.contextualization), 190
- DIAGRAMS (cognite.client.data_classes.contextualization.Contextualization attribute), 185
- disable_gzip (cognite.client.config.GlobalConfig attribute), 11
- disable_ssl (cognite.client.config.GlobalConfig attribute), 11
- DISTRIBUTED (cognite.client.data_classes.contextualization.JobStatus attribute), 197
- DISTRIBUTING (cognite.client.data_classes.contextualization.JobStatus attribute), 197
- download() (cognite.client._api.files.FilesAPI method), 57
- download_bytes() (cognite.client._api.files.FilesAPI method), 58
- download_to_path() (cognite.client._api.files.FilesAPI method), 58
- dump() (cognite.client.data_classes._base.CogniteFilter method), 289
- dump() (cognite.client.data_classes._base.CogniteResource method), 288
- dump() (cognite.client.data_classes._base.CogniteResourceList method), 289
- dump() (cognite.client.data_classes._base.CogniteResponse method), 289
- dump() (cognite.client.data_classes._base.CogniteUpdate method), 289
- dump() (cognite.client.data_classes.annotation_types.images.AssetLink method), 203
- dump() (cognite.client.data_classes.annotation_types.images.ObjectDetection method), 203
- dump() (cognite.client.data_classes.annotation_types.images.TextRegion method), 204
- dump() (cognite.client.data_classes.annotation_types.primitives.BoundingBox method), 204
- dump() (cognite.client.data_classes.annotation_types.primitives.CdfResource method), 205
- dump() (cognite.client.data_classes.annotation_types.primitives.Point method), 186

- method), 205
- dump () (cognite.client.data_classes.annotation_types.primitive.Polygons) (method), 206
- dump () (cognite.client.data_classes.annotation_types.primitive.Polygons) (method), 206
- dump () (cognite.client.data_classes.annotation_types.primitive.Polygons) (method), 207
- dump () (cognite.client.data_classes.annotations.Annotation) (method), 228
- dump () (cognite.client.data_classes.annotations.AnnotationFilter) (method), 229
- dump () (cognite.client.data_classes.assets.Asset) (method), 27
- dump () (cognite.client.data_classes.assets.AssetFilter) (method), 30
- dump () (cognite.client.data_classes.contextualization.AssetTagDetector) (method), 181
- dump () (cognite.client.data_classes.contextualization.ContextualizationJob) (method), 183
- dump () (cognite.client.data_classes.contextualization.ContextualizationJobList) (method), 184
- dump () (cognite.client.data_classes.contextualization.DiagramConverter) (method), 186
- dump () (cognite.client.data_classes.contextualization.DiagramConverter) (method), 187
- dump () (cognite.client.data_classes.contextualization.DiagramConverter) (method), 188
- dump () (cognite.client.data_classes.contextualization.DiagramConverter) (method), 188
- dump () (cognite.client.data_classes.contextualization.DiagramDetector) (method), 190
- dump () (cognite.client.data_classes.contextualization.DiagramDetector) (method), 190
- dump () (cognite.client.data_classes.contextualization.EntityMatchingModel) (method), 193
- dump () (cognite.client.data_classes.contextualization.EntityMatchingModelList) (method), 194
- dump () (cognite.client.data_classes.contextualization.EntityMatchingModelList) (method), 195
- dump () (cognite.client.data_classes.contextualization.FeatureParameters) (method), 196
- dump () (cognite.client.data_classes.contextualization.IndustrialObjectDetectionParameters) (method), 196
- dump () (cognite.client.data_classes.contextualization.IndustrialObjectDetectionParameters) (method), 197
- dump () (cognite.client.data_classes.contextualization.PersonalProtectiveEquipmentDetectionParameters) (method), 198
- dump () (cognite.client.data_classes.contextualization.TextDetectionParameters) (method), 198
- dump () (cognite.client.data_classes.contextualization.VisionExtractItem) (method), 199
- dump () (cognite.client.data_classes.contextualization.VisionExtractJob) (method), 200
- dump () (cognite.client.data_classes.contextualization.VisionExtractPredictions) (method), 201
- dump () (cognite.client.data_classes.contextualization.VisionJob) (method), 202
- dump () (cognite.client.data_classes.datapoints.Datapoints) (method), 105
- dump () (cognite.client.data_classes.datapoints.DatapointsArray) (method), 106
- dump () (cognite.client.data_classes.datapoints.DatapointsArrayList) (method), 107
- dump () (cognite.client.data_classes.extractionpipelines.ExtractionPipeline) (method), 252
- dump () (cognite.client.data_classes.files.FileMetadataFilter) (method), 63
- dump () (cognite.client.data_classes.geospatial.FeatureTagDetector) (method), 156
- dump () (cognite.client.data_classes.iam.TokenInspection) (method), 240
- dump () (cognite.client.data_classes.labels.LabelFilter) (method), 35
- dump () (cognite.client.data_classes.login.LoginStatus) (method), 15
- dump () (cognite.client.data_classes.relationships.RelationshipFilter) (method), 139
- dump () (cognite.client.data_classes.sequences.SequenceData) (method), 121
- dump () (cognite.client.data_classes.templates.TemplateInstance) (method), 224
- dump () (cognite.client.data_classes.templates.ViewResult) (method), 225
- dump () (cognite.client.data_classes.transformations.SessionDetails) (method), 268
- dump () (cognite.client.data_classes.transformations.Transformation) (method), 271
- dump () (cognite.client.data_classes.transformations.TransformationFilter) (method), 273
- dump () (cognite.client.data_classes.transformations.TransformationPreview) (method), 274
- dump () (cognite.client.data_classes.transformations.TransformationUpdate) (method), 275

E

- dump () (cognite.client.data_classes.events.Event) (method), 41
- dump () (cognite.client.data_classes.contextualization.ContextualizationJobType) (method), 198
- EntityMatchingModel (class in cognite.client.data_classes.contextualization), 191
- EntityMatchingModelList (class in cognite.client.data_classes.contextualization), 194
- EntityMatchingModelUpdate (class in cognite.client.data_classes.contextualization), 195

- 195
- errors (*cognite.client.data_classes.contextualization.DiagramDetectionResults* attribute), 191
- errors (*cognite.client.data_classes.contextualization.VisionExtractionJob* attribute), 200
- Event (class in *cognite.client.data_classes.events*), 41
- EventFilter (class in *cognite.client.data_classes.events*), 42
- EventList (class in *cognite.client.data_classes.events*), 43
- events () (*cognite.client.data_classes.assets.Asset* method), 27
- events () (*cognite.client.data_classes.assets.AssetList* method), 31
- events () (*cognite.client.data_classes.transformations.common.TransformationsDestination* static method), 287
- EventUpdate (class in *cognite.client.data_classes.events*), 43
- extend () (*cognite.client.data_classes._base.CogniteResourceList* method), 289
- extend () (*cognite.client.data_classes.contextualization.ContextualizationJobList* method), 185
- extend () (*cognite.client.data_classes.contextualization.DiagramComparison* method), 188
- extend () (*cognite.client.data_classes.contextualization.EntityMatchingModelList* method), 194
- external_id (*cognite.client.data_classes.annotation_types.primitive.CidResourceRef* attribute), 205
- extract () (*cognite.client._api.vision.VisionAPI* method), 180
- ExtractionPipeline (class in *cognite.client.data_classes.extractionpipelines*), 247
- ExtractionPipelineConfig (class in *cognite.client.data_classes.extractionpipelines*), 248
- ExtractionPipelineConfigRevision (class in *cognite.client.data_classes.extractionpipelines*), 249
- ExtractionPipelineConfigRevisionList (class in *cognite.client.data_classes.extractionpipelines*), 250
- ExtractionPipelineContact (class in *cognite.client.data_classes.extractionpipelines*), 250
- ExtractionPipelineList (class in *cognite.client.data_classes.extractionpipelines*), 251
- ExtractionPipelineRun (class in *cognite.client.data_classes.extractionpipelines*), 251
- ExtractionPipelineRunFilter (class in *cognite.client.data_classes.extractionpipelines*), 252
- ExtractionPipelineRunList (class in *cognite.client.data_classes.extractionpipelines*), 253
- ExtractionPipelineUpdate (class in *cognite.client.data_classes.extractionpipelines*), 254
- ## F
- FAILED (*cognite.client.data_classes.contextualization.JobStatus* attribute), 197
- Feature (class in *cognite.client.data_classes.geospatial*), 156
- FeatureDestination (class in *cognite.client.data_classes.geospatial*), 156
- FeatureAggregateList (class in *cognite.client.data_classes.geospatial*), 156
- FeatureList (class in *cognite.client.data_classes.geospatial*), 156
- FeatureListPageList (class in *cognite.client.data_classes.contextualization*), 197
- FeatureType (class in *cognite.client.data_classes.geospatial*), 157
- FeatureTypeList (class in *cognite.client.data_classes.geospatial*), 158
- FeatureTypePatch (class in *cognite.client.data_classes.geospatial*), 158
- FeatureTypeUpdate (class in *cognite.client.data_classes.geospatial*), 158
- fetch_results () (*cognite.client.data_classes.contextualization.DetectJobBundle* method), 185
- FileAggregate (class in *cognite.client.data_classes.files*), 60
- FileMetadata (class in *cognite.client.data_classes.files*), 60
- FileMetadataFilter (class in *cognite.client.data_classes.files*), 61
- FileMetadataList (class in *cognite.client.data_classes.files*), 63
- FileMetadataUpdate (class in *cognite.client.data_classes.files*), 63
- FileReference (class in *cognite.client.data_classes.contextualization*), 196
- files () (*cognite.client.data_classes.assets.Asset* method), 27
- files () (*cognite.client.data_classes.assets.AssetList* method), 31
- files () (*cognite.client.data_classes.transformations.common.TransformationsDestination* static method), 287

filter_nodes() (cognite.client._api.three_d.ThreeDRevisionsAPI method), 166
 first() (cognite.client.data_classes.time_series.TimeSeries method), 86
 fit() (cognite.client._api.entity_matching.EntityMatchingAPI method), 174
 from_geopandas() (cognite.client.data_classes.geospatial.FeatureList static method), 156
 Function (class in cognite.client.data_classes.functions), 74
 FunctionCall (class in cognite.client.data_classes.functions), 76
 FunctionCallList (class in cognite.client.data_classes.functions), 77
 FunctionCallLog (class in cognite.client.data_classes.functions), 77
 FunctionCallLogEntry (class in cognite.client.data_classes.functions), 77
 FunctionList (class in cognite.client.data_classes.functions), 77
 FunctionSchedule (class in cognite.client.data_classes.functions), 77
 FunctionSchedulesList (class in cognite.client.data_classes.functions), 78
 FunctionsLimits (class in cognite.client.data_classes.functions), 78
 FunctionsStatus (class in cognite.client.data_classes.functions), 79
G
 GeospatialComputedItem (class in cognite.client.data_classes.geospatial), 158
 GeospatialComputedItemList (class in cognite.client.data_classes.geospatial), 159
 GeospatialComputedResponse (class in cognite.client.data_classes.geospatial), 159
 GeospatialComputeFunction (class in cognite.client.data_classes.geospatial), 158
 GeospatialGeometryComputeFunction (class in cognite.client.data_classes.geospatial), 159
 GeospatialGeometryTransformComputeFunction (class in cognite.client.data_classes.geospatial), 159
 GeospatialGeometryValueComputeFunction (class in cognite.client.data_classes.geospatial), 160
 get() (cognite.client.CogniteClient method), 10
 get() (cognite.client.data_classes._base.CogniteResourceList method), 289
 get() (cognite.client.data_classes.contextualization.ContextualizationJobList method), 185
 get() (cognite.client.data_classes.contextualization.DiagramConvertPage method), 188
 get() (cognite.client.data_classes.contextualization.EntityMatchingMode method), 194
 get() (cognite.client.data_classes.datapoints.DatapointsArrayList method), 107
 get() (cognite.client.data_classes.datapoints.DatapointsList method), 108
 get_column() (cognite.client.data_classes.sequences.SequenceData method), 121
 get_coordinate_reference_systems() (cognite.client._api.geospatial.GeospatialAPI method), 151
 get_extract_job() (cognite.client._api.vision.VisionAPI method), 181
 get_input_data() (cognite.client.data_classes.functions.FunctionSchedule method), 78
 get_logs() (cognite.client._api.functions.FunctionCallsAPI method), 71
 get_logs() (cognite.client.data_classes.functions.FunctionCall method), 76
 get_raster() (cognite.client._api.geospatial.GeospatialAPI method), 154
 get_response() (cognite.client._api.functions.FunctionCallsAPI method), 70
 get_response() (cognite.client.data_classes.functions.FunctionCall method), 76
 GlobalConfig (class in cognite.client.config), 11
 graphql_query() (cognite.client._api.templates.TemplatesAPI method), 211
 Group (class in cognite.client.data_classes.iam), 238
 groupby_parent_xid() (cognite.client.data_classes.assets.AssetHierarchy method), 31
 GroupList (class in cognite.client.data_classes.iam), 238
 |
 id (cognite.client.data_classes.annotation_types.primitives.CdfResourceReference attribute), 205
 index() (cognite.client.data_classes.contextualization.ContextualizationJobList method), 185
 index() (cognite.client.data_classes.contextualization.DiagramConvertPage method), 188
 index() (cognite.client.data_classes.contextualization.EntityMatchingMode method), 194

INDUSTRIAL_OBJECT_DETECTION (cog- is_not_finished() (cog-
nite.client.data_classes.contextualization.VisionFeature nite.client.data_classes.contextualization.JobStatus
attribute), 202 method), 197

industrial_object_detection_parameters items (cognite.client.data_classes.contextualization.DiagramConvertResul
(cognite.client.data_classes.contextualization.FeatureParameter attribute), 189
attribute), 196 items (cognite.client.data_classes.contextualization.DiagramDetectResult
attribute), 191

industrial_object_predictions (cog- attribute), 191
nite.client.data_classes.contextualization.VisionExtractParameters (cognite.client.data_classes.contextualization.VisionExtractJob
attribute), 201 attribute), 200

IndustrialObjectDetectionParameters items () (cognite.client.data_classes.sequences.SequenceData
(class in cog- method), 121
nite.client.data_classes.contextualization),
196

J

insert () (cognite.client._api.datapoints.DatapointsAPI JobStatus (class in cog-
method), 100 nite.client.data_classes.contextualization),
insert () (cognite.client._api.raw.RawRowsAPI 197
method), 127

insert () (cognite.client._api.sequences.SequencesDataAPI
method), 117

insert () (cognite.client.data_classes.contextualization.ContextualizationJobList Label (class in cognite.client.data_classes.labels), 33
method), 185 LabelFilter (class in cog-
nite.client.data_classes.labels), 33

insert () (cognite.client.data_classes.contextualization.DiagramConvertPageList LabelSearchFilter (class in cog-
method), 188 nite.client.data_classes.labels), 34

insert () (cognite.client.data_classes.contextualization.EntityMatchingModelList LabelSearchFilter (class in cog-
method), 194 nite.client.data_classes.labels), 34

insert_dataframe () (cog- LabelFilter (class in cog-
nite.client._api.datapoints.DatapointsAPI nite.client.data_classes.labels), 34
method), 101

insert_dataframe () (cog- labels () (cognite.client.data_classes.transformations.common.Transform
nite.client._api.raw.RawRowsAPI method), static method), 287
129 latest () (cognite.client.data_classes.time_series.TimeSeries
method), 86

insert_dataframe () (cog- LatestDatapointQuery (class in cog-
nite.client._api.sequences.SequencesDataAPI nite.client.data_classes.datapoints), 108
method), 118

insert_multiple () (cog- list () (cognite.client._api.annotations.AnnotationsAPI
nite.client._api.datapoints.DatapointsAPI method), 226
method), 101 list () (cognite.client._api.assets.AssetsAPI method),
17

inspect () (cognite.client._api.iam.TokenAPI method), list () (cognite.client._api.data_sets.DataSetsAPI
230 method), 45

instance_edges () (cog- list () (cognite.client._api.entity_matching.EntityMatchingAPI
nite.client.data_classes.transformations.common.TransformationDestination method), 194
static method), 287 list () (cognite.client._api.events.EventsAPI method),
36

instance_nodes () (cog- list () (cognite.client._api.extractionpipelines.ExtractionPipelineConfig
nite.client.data_classes.transformations.common.TransformationDestination method), 245
static method), 287 list () (cognite.client._api.extractionpipelines.ExtractionPipelineRunsAPI
method), 243

InstanceEdges (class in cog- list () (cognite.client._api.extractionpipelines.ExtractionPipelinesAPI
nite.client.data_classes.transformations.common), method), 240
285 list () (cognite.client._api.files.FilesAPI method), 50

InstanceNodes (class in cog- list () (cognite.client._api.functions.FunctionCallsAPI
nite.client.data_classes.transformations.common), method), 69
285 list () (cognite.client._api.functions.FunctionsAPI
method), 66

is_finished () (cog- list () (cognite.client._api.functions.FunctionsAPI
nite.client.data_classes.contextualization.JobStatus method), 66
method), 197

- list () (*cognite.client._api.functions.FunctionSchedulesAPI method*), 72
 - list () (*cognite.client._api.geospatial.GeospatialAPI method*), 151
 - list () (*cognite.client._api.iam.APIKeysAPI method*), 231
 - list () (*cognite.client._api.iam.GroupsAPI method*), 233
 - list () (*cognite.client._api.iam.SecurityCategoriesAPI method*), 235
 - list () (*cognite.client._api.iam.ServiceAccountsAPI method*), 230
 - list () (*cognite.client._api.iam.SessionsAPI method*), 236
 - list () (*cognite.client._api.labels.LabelsAPI method*), 32
 - list () (*cognite.client._api.raw.RawDatabasesAPI method*), 123
 - list () (*cognite.client._api.raw.RawRowsAPI method*), 126
 - list () (*cognite.client._api.raw.RawTablesAPI method*), 124
 - list () (*cognite.client._api.relationships.RelationshipsAPI method*), 132
 - list () (*cognite.client._api.sequences.SequencesAPI method*), 110
 - list () (*cognite.client._api.templates.TemplateGroupsAPI method*), 209
 - list () (*cognite.client._api.templates.TemplateGroupVersionsAPI method*), 210
 - list () (*cognite.client._api.templates.TemplateInstancesAPI method*), 215
 - list () (*cognite.client._api.templates.TemplateViewsAPI method*), 218
 - list () (*cognite.client._api.three_d.ThreeDAssetMappingAPI method*), 169
 - list () (*cognite.client._api.three_d.ThreeDModelsAPI method*), 161
 - list () (*cognite.client._api.three_d.ThreeDRevisionsAPI method*), 164
 - list () (*cognite.client._api.time_series.TimeSeriesAPI method*), 80
 - list () (*cognite.client._api.transformations.jobs.TransformationJobsAPI method*), 267
 - list () (*cognite.client._api.transformations.notifications.TransformationNotificationsAPI method*), 265
 - list () (*cognite.client._api.transformations.schedules.TransformationSchedulesAPI method*), 263
 - list () (*cognite.client._api.transformations.TransformationAPI method*), 259
 - list_ancestor_nodes () (*cognite.client._api.three_d.ThreeDRevisionsAPI method*), 167
 - list_calls () (*cognite.client.data_classes.functions.Function method*), 75
 - list_coordinate_reference_systems () (*cognite.client._api.geospatial.GeospatialAPI method*), 151
 - list_feature_types () (*cognite.client._api.geospatial.GeospatialAPI method*), 141
 - list_features () (*cognite.client._api.geospatial.GeospatialAPI method*), 145
 - list_nodes () (*cognite.client._api.three_d.ThreeDRevisionsAPI method*), 166
 - list_schedules () (*cognite.client.data_classes.functions.Function method*), 75
 - list_service_accounts () (*cognite.client._api.iam.GroupsAPI method*), 234
 - LoginStatus (class in *cognite.client.data_classes.login*), 15
- ### M
- max_connection_pool_size (*cognite.client.config.GlobalConfig attribute*), 11
 - max_retries (*cognite.client.config.GlobalConfig attribute*), 11
 - max_retries_connect (*cognite.client.config.GlobalConfig attribute*), 11
 - max_retry_backoff (*cognite.client.config.GlobalConfig attribute*), 11
 - metrics () (*cognite.client.data_classes.transformations.jobs.TransformationJobsAPI method*), 280
 - monkeypatch_cognite_client () (in module *cognite.client.testing*), 293
 - ms_to_datetime () (in module *cognite.client.utils*), 293
- ### N
- name (*cognite.client.data_classes.contextualization.EntityMatchingModel attribute*), 105
 - nan_to_none () (in module *cognite.client.data_classes.geospatial*), 160
- ### O
- OAuthClientCertificate (class in *cognite.client.credentials*), 13
 - OAuthClientCredentials (class in *cognite.client.credentials*), 12
 - OAuthDeviceCode (class in *cognite.client.credentials*), 13

OAuthInteractive (class in cog-nite.client.credentials), 13

ObjectDetection (class in cog-nite.client.data_classes.annotation_types.images), 203

OrderSpec (class in cog-nite.client.data_classes.geospatial), 160

P

pages (cognite.client.data_classes.contextualization.DiagramConvertPage attribute), 186

parent () (cognite.client.data_classes.assets.Asset method), 27

partial_match (cog-nite.client.data_classes.contextualization.AssetTagDetectionParameters attribute), 181

patch_feature_types () (cog-nite.client._api.geospatial.GeospatialAPI method), 142

Patches (class in cog-nite.client.data_classes.geospatial), 160

PEOPLE_DETECTION (cog-nite.client.data_classes.contextualization.VisionFeature attribute), 202

people_detection_parameters (cog-nite.client.data_classes.contextualization.FeatureParameters attribute), 196

people_predictions (cog-nite.client.data_classes.contextualization.VisionExtractPredictions attribute), 201

PeopleDetectionParameters (class in cog-nite.client.data_classes.contextualization), 197

PERSONAL_PROTECTIVE_EQUIPMENT_DETECTION (cognite.client.data_classes.contextualization.VisionFeature attribute), 202

Q

personal_protective_equipment_detection_parameters (cognite.client.data_classes.contextualization.FeatureParameters attribute), 196

personal_protective_equipment_predictions (cognite.client.data_classes.contextualization.VisionExtractPredictions attribute), 201

PersonalProtectiveEquipmentDetectionParameters (class in cog-nite.client.data_classes.contextualization), 198

Point (class in cog-nite.client.data_classes.annotation_types.primitives), 205

Polygon (class in cog-nite.client.data_classes.annotation_types.primitives), 206

polygon (cognite.client.data_classes.annotation_types.images.ObjectDetection attribute), 203

PolyLine (class in cog-nite.client.data_classes.annotation_types.primitives), 206

polyline (cognite.client.data_classes.annotation_types.images.ObjectDetection attribute), 203

pop () (cognite.client.data_classes.contextualization.ContextualizationJob method), 185

pop () (cognite.client.data_classes.contextualization.DiagramConvertPage method), 188

pop () (cognite.client.data_classes.contextualization.EntityMatchingMode method), 195

post () (cognite.client.CogniteClient method), 10

predict () (cognite.client._api.entity_matching.EntityMatchingAPI method), 177

predict () (cognite.client.data_classes.contextualization.EntityMatchingMode method), 193

preview () (cognite.client._api.transformations.TransformationsAPI method), 257

ProjectSpec (class in cog-nite.client.data_classes.iam), 238

PropertyAndSearchSpec (class in cog-nite.client.data_classes.geospatial), 160

proxies (cognite.client.config.GlobalConfig attribute), 12

put () (cognite.client.CogniteClient method), 10

put_raster () (cog-nite.client._api.geospatial.GeospatialAPI method), 152

R

raw () (cognite.client.data_classes.transformations.common.Transformation static method), 287

raw_resolver (class in cog-nite.client.data_classes.templates), 219

raw_table (class in cog-nite.client.data_classes.transformations.common), 285

refit () (cognite.client._api.entity_matching.EntityMatchingAPI method), 175

refit () (cognite.client.data_classes.contextualization.EntityMatchingMode method), 193

Relationship (class in cog-nite.client.data_classes.relationships), 136

RelationshipFilter (class in cog-nite.client.data_classes.relationships), 137

RelationshipList (class in `cognite.client.data_classes.relationships`), 139

relationships() (cognite.client.data_classes.transformations.common.TransformationsAPI static method), 288

RelationshipUpdate (class in `cognite.client.data_classes.relationships`), 140

remove() (cognite.client.data_classes.contextualization.ContextualizationJobList method), 185

remove() (cognite.client.data_classes.contextualization.DiagramConversionPageList method), 188

remove() (cognite.client.data_classes.contextualization.EntityMatchingModelList method), 195

remove_service_account() (cognite.client._api.iam.GroupsAPI method), 235

resolve() (cognite.client._api.templates.TemplateViewsAPI method), 218

result (cognite.client.data_classes.contextualization.ContextualizationJob cognite.client._api.transformations.schema.TransformationJobList attribute), 184

result (cognite.client.data_classes.contextualization.DetectJobBundle() (cognite.client._api.transformations.TransformationsAPI attribute), 185

result (cognite.client.data_classes.contextualization.DiagramConversionResults() (cognite.client._api.datapoints.DatapointsAPI attribute), 189

result (cognite.client.data_classes.contextualization.DiagramDetectionResult, 93 attribute), 191

result (cognite.client.data_classes.contextualization.VisionExtractJob cognite.client.data_classes.functions.Function attribute), 200

result (cognite.client.data_classes.contextualization.VisionJob cognite.client._api.datapoints.DatapointsAPI attribute), 202

retrieve() (cognite.client._api.annotations.AnnotationsAPI method), 226

retrieve() (cognite.client._api.assets.AssetsAPI method), 15

retrieve() (cognite.client._api.data_sets.DataSetsAPI method), 44

retrieve() (cognite.client._api.datapoints.DatapointsAPI method), 89

retrieve() (cognite.client._api.entity_matching.EntityMatchingAPI method), 175

retrieve() (cognite.client._api.events.EventsAPI method), 35

retrieve() (cognite.client._api.extractionpipelines.ExtractionPipelineConfigsAPI method), 245

retrieve() (cognite.client._api.extractionpipelines.ExtractionPipelinesAPI method), 241

retrieve() (cognite.client._api.files.FilesAPI method), 49

retrieve() (cognite.client._api.functions.FunctionCallsAPI method), 70

retrieve() (cognite.client._api.functions.FunctionsAPI method), 67

retrieve() (cognite.client._api.raw.RawRowsAPI method), 126

retrieve() (cognite.client._api.relationships.RelationshipsAPI method), 131

retrieve() (cognite.client._api.sequences.SequencesAPI method), 108

retrieve() (cognite.client._api.sequences.SequencesDataAPI method), 115

retrieve() (cognite.client._api.three_d.ThreeDFilesAPI method), 115

retrieve() (cognite.client._api.three_d.ThreeDModelsAPI method), 115

retrieve() (cognite.client._api.three_d.ThreeDRevisionsAPI method), 115

retrieve() (cognite.client._api.time_series.TimeSeriesAPI method), 79

retrieve() (cognite.client._api.transformations.jobs.TransformationJobList method), 266

retrieve() (cognite.client._api.transformations.schedules.TransformationScheduleList method), 261

retrieve() (cognite.client._api.transformations.schema.TransformationJobList method), 267

retrieve() (cognite.client._api.transformations.TransformationsAPI method), 255

retrieve_dataframes() (cognite.client._api.datapoints.DatapointsAPI method), 95

retrieve_dataframe() (cognite.client._api.raw.RawRowsAPI method), 128

retrieve_dataframe() (cognite.client._api.sequences.SequencesDataAPI method), 116

retrieve_dataframe_in_tz() (cognite.client._api.datapoints.DatapointsAPI method), 96

retrieve_download_urls() (cognite.client._api.files.FilesAPI method), 57

retrieve_types() (cognite.client._api.geospatial.GeospatialAPI method), 141

retrieve_features() (cognite.client._api.geospatial.GeospatialAPI method), 144

retrieve_latest() (cognite.client._api.datapoints.DatapointsAPI method), 98

retrieve_multiple() (cognite.client._api.annotations.AnnotationsAPI method), 226

retrieve_multiple() (cognite.client._api.assets.AssetsAPI method), 16
retrieve_multiple() (cognite.client._api.data_sets.DataSetsAPI method), 44
retrieve_multiple() (cognite.client._api.entity_matching.EntityMatchingAPI method), 176
retrieve_multiple() (cognite.client._api.events.EventsAPI method), 36
retrieve_multiple() (cognite.client._api.extractionpipelines.ExtractionPipelinesAPI method), 242
retrieve_multiple() (cognite.client._api.files.FilesAPI method), 50
retrieve_multiple() (cognite.client._api.functions.FunctionsAPI method), 68
retrieve_multiple() (cognite.client._api.relationships.RelationshipsAPI method), 131
retrieve_multiple() (cognite.client._api.sequences.SequencesAPI method), 110
retrieve_multiple() (cognite.client._api.templates.TemplateGroupsAPI method), 208
retrieve_multiple() (cognite.client._api.templates.TemplateInstancesAPI method), 214
retrieve_multiple() (cognite.client._api.time_series.TimeSeriesAPI method), 79
retrieve_multiple() (cognite.client._api.transformations.jobs.TransformationJobsAPI method), 266
retrieve_multiple() (cognite.client._api.transformations.schedules.TransformationSchedulesAPI method), 262
retrieve_multiple() (cognite.client._api.transformations.TransformationsAPI method), 256
retrieve_subtree() (cognite.client._api.assets.AssetsAPI method), 16
reverse() (cognite.client.data_classes.contextualization.ContextualizationJobList method), 185
reverse() (cognite.client.data_classes.contextualization.DiagramConverterPageList method), 188
reverse() (cognite.client.data_classes.contextualization.EntityMatchingModelList method), 195
revert() (cognite.client._api.extractionpipelines.ExtractionPipelineConfigsAPI method), 246
RevisionCameraProperties (class in cognite.client.data_classes.three_d), 170
revoke() (cognite.client._api.iam.SessionsAPI method), 237
Row (class in cognite.client.data_classes.raw), 130
RowList (class in cognite.client.data_classes.raw), 130
Rows () (cognite.client.data_classes.raw.Table method), 131
run() (cognite.client._api.transformations.TransformationsAPI method), 256
run_async() (cognite.client._api.transformations.TransformationsAPI method), 257
RUNNING (cognite.client.data_classes.contextualization.JobStatus attribute), 197

S

save_predictions () (cognite.client.data_classes.contextualization.VisionExtractJob method), 200
search () (cognite.client._api.assets.AssetsAPI method), 20
search () (cognite.client._api.events.EventsAPI method), 39
search () (cognite.client._api.files.FilesAPI method), 53
search () (cognite.client._api.sequences.SequencesAPI method), 112
search () (cognite.client._api.time_series.TimeSeriesAPI method), 82
search_features () (cognite.client._api.geospatial.GeospatialAPI method), 147
SecurityCategory (class in cognite.client.data_classes.iam), 238
SecurityCategoryList (class in cognite.client.data_classes.iam), 239
sequence_rows () (cognite.client.data_classes.transformations.common.TransformationsAPI method), 288
SequenceAggregate (class in cognite.client.data_classes.sequences), 120
SequenceColumnUpdate (class in cognite.client.data_classes.sequences), 120
SequenceData (class in cognite.client.data_classes.sequences), 120
SequenceDataList (class in cognite.client.data_classes.sequences), 121
SequenceFilter (class in cognite.client.data_classes.sequences), 122
SequenceList (class in cognite.client.data_classes.sequences), 122
SequenceRows (class in cognite.client.data_classes.transformations.common), 122

286

sequences() (*cognite.client.data_classes.assets.Asset* method), 28

sequences() (*cognite.client.data_classes.assets.AssetList* method), 31

sequences() (*cognite.client.data_classes.transformations.common.TransformationsDestinationTemplates* static method), 288

SequenceUpdate (class in *cognite.client.data_classes.sequences*), 123

ServiceAccount (class in *cognite.client.data_classes.iam*), 239

ServiceAccountList (class in *cognite.client.data_classes.iam*), 239

Session (class in *cognite.client.data_classes.iam*), 239

SessionDetails (class in *cognite.client.data_classes.transformations*), 268

SessionList (class in *cognite.client.data_classes.iam*), 240

sort() (*cognite.client.data_classes.contextualization.ContextualizationJobList* method), 185

sort() (*cognite.client.data_classes.contextualization.DiagramConversionPackageList* method), 188

sort() (*cognite.client.data_classes.contextualization.EntityMatchingModelList* method), 195

Source (class in *cognite.client.data_classes.templates*), 220

status() (*cognite.client._api.login.LoginAPI* method), 14

status_forcelist (*cognite.client.config.GlobalConfig* attribute), 11

stream_features() (*cognite.client._api.geospatial.GeospatialAPI* method), 149

string_datapoints() (*cognite.client.data_classes.transformations.common.TransformationsDestinationTemplates* static method), 288

StringFilter (class in *cognite.client.data_classes.extractionpipelines*), 254

subtree() (*cognite.client.data_classes.assets.Asset* method), 28

suggest() (*cognite.client._api.annotations.AnnotationsAPI* method), 227

SyntheticTimeSeriesResolver (class in *cognite.client.data_classes.templates*), 220

TemplateGroup (class in *cognite.client.data_classes.templates*), 222

TemplateGroupList (class in *cognite.client.data_classes.templates*), 222

TemplateGroupVersion (class in *cognite.client.data_classes.templates*), 222

TemplateGroupVersionList (class in *cognite.client.data_classes.templates*), 223

TemplateInstance (class in *cognite.client.data_classes.templates*), 223

TemplateInstanceList (class in *cognite.client.data_classes.templates*), 224

TemplateInstanceUpdate (class in *cognite.client.data_classes.templates*), 224

TEXT_DETECTION (*cognite.client.data_classes.contextualization.VisionFeature* attribute), 202

text_detection_parameters (*cognite.client.data_classes.contextualization.FeatureParameters* attribute), 196

text_predictions (*cognite.client.data_classes.contextualization.VisionExtractPredictionParameters* attribute), 201

TextRegion (class in *cognite.client.data_classes.annotation_types.images*), 204

ThreeDAssetMapping (class in *cognite.client.data_classes.three_d*), 170

ThreeDAssetMappingList (class in *cognite.client.data_classes.three_d*), 171

ThreeDModel (class in *cognite.client.data_classes.three_d*), 171

ThreeDModelList (class in *cognite.client.data_classes.three_d*), 171

ThreeDModelRevision (class in *cognite.client.data_classes.three_d*), 172

ThreeDModelRevisionList (class in *cognite.client.data_classes.three_d*), 173

ThreeDModelRevisionUpdate (class in *cognite.client.data_classes.three_d*), 173

ThreeDModelUpdate (class in *cognite.client.data_classes.three_d*), 173

ThreeDNode (class in *cognite.client.data_classes.three_d*), 173

ThreeDNodeList (class in *cognite.client.data_classes.three_d*), 174

threshold (*cognite.client.data_classes.contextualization.AssetTagDetection* attribute), 181

threshold (*cognite.client.data_classes.contextualization.IndustrialObjectDetection* attribute), 197

threshold (*cognite.client.data_classes.contextualization.PeopleDetection* attribute), 197

T

Table (class in *cognite.client.data_classes.raw*), 130

TableList (class in *cognite.client.data_classes.raw*), 131

tables() (*cognite.client.data_classes.raw.Database* method), 130

- attribute), 197
- threshold(*cognite.client.data_classes.contextualization.PersonalProtectionEquipmentDetectionParameters* attribute), 198
- threshold(*cognite.client.data_classes.contextualization.TextDetectionParameters* attribute), 198
- threshold(*cognite.client.data_classes.contextualization.ThresholdParameter* attribute), 199
- ThresholdParameter (class in *cognite.client.data_classes.contextualization*), 199
- time_series() (*cognite.client.data_classes.assets.Asset* method), 28
- time_series() (*cognite.client.data_classes.assets.AssetList* method), 31
- TimeSeries (class in *cognite.client.data_classes.time_series*), 85
- timeseries() (*cognite.client.data_classes.transformations.common.TransformationsDestination* static method), 288
- TimeSeriesAggregate (class in *cognite.client.data_classes.time_series*), 86
- TimeSeriesFilter (class in *cognite.client.data_classes.time_series*), 86
- TimeSeriesList (class in *cognite.client.data_classes.time_series*), 88
- TimeSeriesUpdate (class in *cognite.client.data_classes.time_series*), 88
- timestamp_to_ms() (in module *cognite.client.utils*), 293
- to_api_item() (*cognite.client.data_classes.contextualization.FileReference* method), 196
- to_geopandas() (*cognite.client.data_classes.geospatial.FeatureList* method), 157
- to_json_payload() (*cognite.client.data_classes.geospatial.GeospatialComputerFunction* method), 158
- to_json_payload() (*cognite.client.data_classes.geospatial.GeospatialGeometryTransformationFunction* method), 159
- to_json_payload() (*cognite.client.data_classes.geospatial.GeospatialGeometryValueComputerFunction* method), 160
- to_pandas() (*cognite.client.data_classes._base.CogniteResponse* method), 288
- to_pandas() (*cognite.client.data_classes._base.CogniteResponseList* method), 289
- to_pandas() (*cognite.client.data_classes.annotation_types.images.AssetLink* method), 203
- to_pandas() (*cognite.client.data_classes.annotation_types.images.ObjectDetection* method), 204
- to_pandas() (*cognite.client.data_classes.annotation_types.images.TextDetection* method), 204
- to_pandas() (*cognite.client.data_classes.annotation_types.images.EquipmentDetectionParameters* method), 204
- to_pandas() (*cognite.client.data_classes.annotation_types.primitives.Boolean* method), 204
- to_pandas() (*cognite.client.data_classes.annotation_types.primitives.Character* method), 205
- to_pandas() (*cognite.client.data_classes.annotation_types.primitives.Date* method), 205
- to_pandas() (*cognite.client.data_classes.annotation_types.primitives.Float* method), 205
- to_pandas() (*cognite.client.data_classes.annotation_types.primitives.Integer* method), 206
- to_pandas() (*cognite.client.data_classes.annotation_types.primitives.String* method), 206
- to_pandas() (*cognite.client.data_classes.assets.Asset* method), 28
- to_pandas() (*cognite.client.data_classes.contextualization.AssetId* method), 182
- to_pandas() (*cognite.client.data_classes.contextualization.ContextualizationDestination* method), 185
- to_pandas() (*cognite.client.data_classes.contextualization.DiagramComponent* method), 186
- to_pandas() (*cognite.client.data_classes.contextualization.DiagramComponentList* method), 187
- to_pandas() (*cognite.client.data_classes.contextualization.DiagramComponentList* method), 188
- to_pandas() (*cognite.client.data_classes.contextualization.DiagramComponentList* method), 189
- to_pandas() (*cognite.client.data_classes.contextualization.DiagramComponentList* method), 190
- to_pandas() (*cognite.client.data_classes.contextualization.DiagramComponentList* method), 191
- to_pandas() (*cognite.client.data_classes.contextualization.EntityMatch* method), 193
- to_pandas() (*cognite.client.data_classes.contextualization.EntityMatch* method), 195
- to_pandas() (*cognite.client.data_classes.contextualization.FeatureParameter* method), 196
- to_pandas() (*cognite.client.data_classes.contextualization.IndustrialObject* method), 197
- to_pandas() (*cognite.client.data_classes.contextualization.PeopleDetection* method), 197
- to_pandas() (*cognite.client.data_classes.contextualization.PersonalProtectionEquipment* method), 198
- to_pandas() (*cognite.client.data_classes.contextualization.TextDetectionParameters* method), 199
- to_pandas() (*cognite.client.data_classes.contextualization.VisionExtract* method), 199
- to_pandas() (*cognite.client.data_classes.contextualization.VisionExtract* method), 200
- to_pandas() (*cognite.client.data_classes.contextualization.VisionExtract* method), 201

to_pandas() (*cognite.client.data_classes.contextualization.VisionData* method), 202

TransformationList (class in *cognite.client.data_classes.transformations*), 273

to_pandas() (*cognite.client.data_classes.datapoints.Datapoint* method), 104

TransformationNotification (class in *cognite.client.data_classes.transformations.notifications*), 276

to_pandas() (*cognite.client.data_classes.datapoints.DatapointsTransformationNotificationFilter* method), 105

TransformationNotificationFilter (class in *cognite.client.data_classes.transformations.notifications*), 276

to_pandas() (*cognite.client.data_classes.datapoints.DatapointsTransformationNotificationList* method), 108

TransformationNotificationList (class in *cognite.client.data_classes.transformations.notifications*), 278

to_pandas() (*cognite.client.data_classes.datapoints.DatapointsTransformationNotificationList* method), 108

TransformationNotificationList (class in *cognite.client.data_classes.transformations.notifications*), 278

to_pandas() (*cognite.client.data_classes.login.LoginStatus* method), 15

TransformationPreviewResult (class in *cognite.client.data_classes.transformations*), 273

to_pandas() (*cognite.client.data_classes.raw.Row* method), 130

TransformationSchedule (class in *cognite.client.data_classes.transformations.schedules*), 275

to_pandas() (*cognite.client.data_classes.raw.RowList* method), 130

TransformationScheduleList (class in *cognite.client.data_classes.transformations.schedules*), 276

to_pandas() (*cognite.client.data_classes.sequences.SequenceDataTransformationScheduleList* method), 121

TransformationScheduleUpdate (class in *cognite.client.data_classes.transformations.schedules*), 276

Token (class in *cognite.client.credentials*), 12

TransformationSchemaColumn (class in *cognite.client.data_classes.transformations.schema*), 283

TokenInspection (class in *cognite.client.data_classes.iam*), 240

TransformationSchemaColumnList (class in *cognite.client.data_classes.transformations.schema*), 284

Transformation (class in *cognite.client.data_classes.transformations*), 268

TransformationUpdate (class in *cognite.client.data_classes.transformations*), 274

TransformationBlockedInfo (class in *cognite.client.data_classes.transformations.common*), 286

TransformationJob (class in *cognite.client.data_classes.transformations.jobs*), 278

TransformationJobFilter (class in *cognite.client.data_classes.transformations.jobs*), 282

TransformationJobList (class in *cognite.client.data_classes.transformations.jobs*), 282

TransformationJobMetric (class in *cognite.client.data_classes.transformations.jobs*), 282

TransformationJobMetricList (class in *cognite.client.data_classes.transformations.jobs*), 283

TransformationJobStatus (class in *cognite.client.data_classes.transformations.jobs*), 283

U

update() (*cognite.client._api.annotations.AnnotationsAPI* method), 227

update() (*cognite.client._api.assets.AssetsAPI* method), 25

update() (*cognite.client._api.data_sets.DataSetsAPI* method), 47

update() (*cognite.client._api.entity_matching.EntityMatchingAPI* method), 177

update() (*cognite.client._api.events.EventsAPI* method), 40

update() (*cognite.client._api.extractionpipelines.ExtractionPipelinesAPI* method), 242

update() (*cognite.client._api.files.FilesAPI* method), 59

update() (*cognite.client._api.relationships.RelationshipsAPI* method), 134

update () (*cognite.client._api.sequences.SequencesAPI* method), 114
 update () (*cognite.client._api.templates.TemplateInstancesAPI* method), 214
 update () (*cognite.client._api.three_d.ThreeDModelsAPI* method), 162
 update () (*cognite.client._api.three_d.ThreeDRevisionsAPI* method), 164
 update () (*cognite.client._api.time_series.TimeSeriesAPI* method), 84
 update () (*cognite.client._api.transformations.schedules.TransformationSchedulesAPI* method), 263
 update () (*cognite.client._api.transformations.TransformationsAPI* method), 260
 update () (*cognite.client.data_classes.functions.Function* method), 76
 update () (*cognite.client.data_classes.functions.FunctionCall* method), 76
 update () (*cognite.client.data_classes.transformations.jobs.TransformationJob* method), 280
 update_features () (*cognite.client._api.geospatial.GeospatialAPI* method), 145
 update_status () (*cognite.client.data_classes.contextualization.ContextualizationJob* method), 184
 update_status () (*cognite.client.data_classes.contextualization.DiagramConvertResults* method), 189
 update_status () (*cognite.client.data_classes.contextualization.DiagramDetectResults* method), 191
 update_status () (*cognite.client.data_classes.contextualization.EntityMatchingModel* method), 193
 update_status () (*cognite.client.data_classes.contextualization.VisionExtractJob* method), 201
 update_status () (*cognite.client.data_classes.contextualization.VisionJob* method), 202
 update_thumbnail () (*cognite.client._api.three_d.ThreeDRevisionsAPI* method), 165
 upload () (*cognite.client._api.files.FilesAPI* method), 54
 upload_bytes () (*cognite.client._api.files.FilesAPI* method), 56
 upsert () (*cognite.client._api.templates.TemplateGroupsAPI* method), 208
 upsert () (*cognite.client._api.templates.TemplateGroupVersionsAPI* method), 210
 upsert () (*cognite.client._api.templates.TemplateInstancesAPI* method), 213
 upsert () (*cognite.client._api.templates.TemplateViewsAPI* method), 217
 version (*cognite.client.CogniteClient* attribute), 10
 View (class in *cognite.client.data_classes.templates*), 225
 ViewList (class in *cognite.client.data_classes.templates*), 225
 ViewResolveList (class in *cognite.client.data_classes.templates*), 225
 ViewResolver (class in *cognite.client.data_classes.templates*), 225
 VISION (*cognite.client.data_classes.contextualization.ContextualizationJob* attribute), 185
 VisionExtractItem (class in *cognite.client.data_classes.contextualization*), 199
 VisionExtractJob (class in *cognite.client.data_classes.contextualization*), 200
 VisionExtractPredictions (class in *cognite.client.data_classes.contextualization*), 201
 VisionFeature (class in *cognite.client.data_classes.contextualization*), 201
 ConvertResults (class in *cognite.client.data_classes.contextualization*), 189
 DetectResults (class in *cognite.client.data_classes.annotation_types.primitives*), 207
 Wait (class in *cognite.client.data_classes.contextualization*), 193
 wait () (*cognite.client.data_classes.transformations.jobs.TransformationJob* method), 280
 wait_async () (*cognite.client.data_classes.transformations.jobs.TransformationJob* method), 281
 wait_for_completion () (*cognite.client.data_classes.contextualization.ContextualizationJob* method), 184
 wait_for_completion () (*cognite.client.data_classes.contextualization.DetectJobBundle* method), 186
 wait_for_completion () (*cognite.client.data_classes.contextualization.DiagramConvertResults* method), 189
 wait_for_completion () (*cognite.client.data_classes.contextualization.DiagramDetectResults* method), 191
 wait_for_completion () (*cognite.client.data_classes.contextualization.EntityMatchingModel* method), 193

method), 194
wait_for_completion() (*cog-
nite.client.data_classes.contextualization.VisionExtractJob
method*), 201
wait_for_completion() (*cog-
nite.client.data_classes.contextualization.VisionJob
method*), 202