Alphacast Python Documentation

Release 1.0

Alphacast

Oct 06, 2021

CONTENTS

Contents:

CHAPTER

ONE

ALPHACAST PYTHON LIBRARY

1.1 Quick Start

The Alphacast Python library allows you to interact with the assets hosted in Alphacast without the need to know the details of the API integration. In the current version you can interact with your repositories and datasets.

Begin by installing Alphacast SDK in your console

```
pip install alphacast
```

Then import the module in your screen and initialize the session with your API key. To find you API you need to have an Alphacast account (https://www.alphacast.io/api/auth/login). Once created, find your key in your settings (click on your user on the left menu)

```
from alphacast import Alphacast
alphacast = Alphacast(YOUR_API_KEY)
```

1.2 Repositories

All the interaction with your repositories are handled with the "repository" class. To read the metadata of all your repositories and those you have write permissions use repository.read_all()

```
alphacast.repository.read_all()
```

To read the metadata of a single repository by id use read_by_id or read_by_name. You need to have owner, admin or write permissions to access the repo.

```
alphacast.repository.read_by_id(repo_id)
alphacast.repository.read_by_name(repo_name)
```

To create a repository you need to define its name, slug (optional), privacy (optional) and description (also optional). The parameter returnIdIfExists (True/False) is usen to describe the action when de repository already exists. If true, then it returns de Id.

```
alphacast.repository.create("my first Test Repo", repo_description="This is my first Repo
↪", slug="test-repo", privacy="Public", returnIdIfExists=True)
```

1.3 Finding datasets

Finding and downloading data is at the core of Alphacast and is done with the "datasets" class.

To access the metadata of all your datasets (that is, those where you have owner, admin or write permission) use read_all() or read_by_name().

```
alphacast.datasets.read_all()
alphacast.datasets.read_by_name("dataset_name")
```

NOTE Only your dataset can be accessed with these methods. If you want to use public data you have to use the id which can be found in the url of the dataset in alphacast.io web.

With the id you can also access information of datasets where you have read permission (either because they are public or because you have been granted access)

use the dataset(dataset_id).metadata() and datestats() methods to access more information.

metadata retrieves the values of id, name, createdAt, updatedAt, repositoryId and permission levels.

datestats retrieves the infered Frequency and the first and last Date available

```
alphacast.datasets.dataset(5565).metadata()
alphacast.datasets.dataset(5208).datestats()
```

1.4 Downloading data

The method download_data() of the Class dataset() is used to retrieve the data from the datasets. You need to have read permission (or above) to access the data

```
# for json/xlsx/csv data use format = "json"  / "xlsx" / "csv"
json_data = alphacast.datasets.dataset(6755).download_data(format = "json")
excel_file = alphacast.datasets.dataset(6755).download_data("xlsx")
csv_data = alphacast.datasets.dataset(6755).download_data("csv")

# To load this into a Pandas dataframe
import pandas as pd
import io
df = pd.read_csv( io.StringIO(alphacast.datasets.dataset(6755).download_data("csv").
↪decode("UTF-8")))

# or directly
df = alphacast.datasets.dataset(6755).download_data("pandas")
```

1.5 Creating datasets

Creating datasets and uploading information is a two step process. First you need to create the datasets and "initialize" its columns. We need to know which are the "Date" and the Entity column or columns.

Entity can be defined as one or many columns as long as the pairs of Date / Entity are unique. Basically, think of Date / Entity as a unique index.

Important Note If you want to create Alphacast charts with your data then Entity need to be a single columns (Date / Entity pair). Our chart engine accept, for the moment, only single entity datasets.

So first let's create a dataset

```
alphacast.datasets.create(dataset_name, repo_id, description)
```

The process, if succesfull, will provide you with an id. you can check if your dataset has been created visiting alphacast.io/datasets/{dataset_id}

1.6 Uploading data

Now let's insert some data into that dataset. We will use the pandas dataframe loaded before. Uploading using Pandas Dataframes is an easy way to do it, but plain csv can be uploaded.

```
# keep some variables from the dataset
df = df[['Date', 'country', 'CPI - All Urban Wage Earners and Clerical Workers - current_
↪prices_yoy']]

# initialize de variables. We will use "Date" as date column and "country" as entity.
alphacast.datasets.dataset({dataset_id}).initialize_columns(dateColumnName = "Date",␣
↪entitiesColumnNames=["country"], dateFormat= "%Y-%m-%d")
```

Response

```
[{"id": {dataset_id}, "columnDefinitions": [{"sourceName": "Date", "dataType":
↪"Date", "dateFormat": "%Y-%m-%d", "isEntity": "True"}, {"sourceName":
↪"country", "isEntity": "True"}], "updateAt": "2021-10-06T16:51:35.418493"}]
```

Next step. Upload the data. Four parameters are needed. "df" is The data and uploadIndex defines if the DataFrame index should be uploaded also.

deleteMissingFromDB and onConflictUpdateDB are two parameters to decide the behaviour of what to do with if there is data already on the dataset. If deleteMissingFromDB is false everything that is not sent in the current upload will be deleted. If onConflictUpdateDB the conflicting values of matching Date / Entities will be updated.

```
alphacast.datasets.dataset(7938).upload_data_from_df(df, deleteMissingFromDB = False,␣
↪onConflictUpdateDB = False, uploadIndex=False)

#upload_data_from_csv() is also available
```

Now head to https://www.alphacast.io/datasets/{dataset_id} to see the result.

1.7 Process status

Your request creates a upload process in Alphacast, that may take some time. You will get the id of that process when submiting the upload. It will look like this

Response


```
[{"id": 45141, "status": "Requested", "createdAt": "2021-10-06T16:58:18.999786
↪", "datasetId": 7938}]
```


To check the status of all your processes for that dataset use

```
alphacast.datasets.dataset(7938).processes()
```

Response


```
[{"id": 45141, "datasetId": 7938, "status": "Processed", "statusDescription":
↪"1292 values added to database./n", "deleteMissingFromDB": 0,
↪"onConflictUpdateDB": 0, "createdAt": "2021-10-06T16:58:18", "processedAt":
↪"2021-10-05T15:40:52"}]
```


or alternatively

```
alphacast.datasets.dataset(7938).process(45141)
```

ok! We are done. Good Job!!

Much more features are coming down the road. Stay tuned. We would love to hear your feedback at hello@alphacast.io