UNFCCC DI API Documentation

Release 2.0.1

Mika Pflüger

Sep 21, 2021

CONTENTS:

1 UNFCCC DI API **1**
- 1.1 Features . . . 1
- 1.2 Citation . . . 1
- 1.3 Data package . . . 1
- 1.4 CI status and other links . . . 1

2 Installation **3**
- 2.1 Stable release . . . 3
- 2.2 From sources . . . 3

3 Usage **5**

4 API **9**
- 4.1 unfccc_di_api . . . 9
- 4.2 unfccc_di_api.tests . . . 11

5 Contributing **13**
- 5.1 Types of Contributions . . . 13
- 5.2 Get Started! . . . 14
- 5.3 Pull Request Guidelines . . . 14
- 5.4 Deploying . . . 15

6 Credits **17**
- 6.1 Developers . . . 17
- 6.2 Contributors . . . 17
- 6.3 Libraries . . . 17

7 Changelog **19**
- 7.1 2.0.1 (2021-04-23) . . . 19
- 7.2 2.0.0 (2021-02-09) . . . 19
- 7.3 1.1.1 (2021-02-08) . . . 19
- 7.4 1.1.0 (2021-01-25) . . . 19
- 7.5 1.0.0 (2021-01-22) . . . 19
- 7.6 0.1.0 (2021-01-22) . . . 20

8 Indices and tables **21**

Python Module Index **23**

Index **25**

CHAPTER

ONE

UNFCCC DI API

Python wrapper around the Flexible Query API of the UNFCCC.

- Free software: Apache Software License 2.0
- Documentation: https://unfccc-di-api.readthedocs.io.

1.1 Features

- High-level API to query all information for a given party.
- Low-level API to selectively query information with the same resolution as in the UNFCCC web query tool.

1.2 Citation

If you use this library and want to cite it, please cite it as:

Mika Pflüger & Daniel Huppmann. (2021-02-09). pik-primap/unfccc_di_api: Version 2.0.1. Zenodo. https://doi.org/10.5281/zenodo.4525036

1.3 Data package

If you just want all the data in CSV and parquet format (suitable for reading with pandas), look at our data package.

1.4 CI status and other links

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install UNFCCC DI API, run this command in your terminal:

```
$ pip install unfccc_di_api
```

This is the preferred method to install UNFCCC DI API, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for UNFCCC DI API can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/pik-primap/unfccc_di_api
```

Or download the tarball:

```
$ curl -OJL https://github.com/pik-primap/unfccc_di_api/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use the UNFCCC API, import the package and instantiate the reader, which will download the most recent metadata from the UNFCCC:

[1]:
```
import unfccc_di_api

reader = unfccc_di_api.UNFCCCApiReader()
```

Check for which parties data is available:

[2]:
```
reader.parties
```

[2]:
```
       code                name noData
id
3       AUS           Australia    NaN
4       AUT             Austria    NaN
5       BEL             Belgium    NaN
6       BGR            Bulgaria    NaN
7       BLR             Belarus    NaN
...     ...                 ...    ...
100214  VNM            Viet Nam    NaN
100215  YEM               Yemen    NaN
100216  ZMB              Zambia    NaN
100217  ZWE            Zimbabwe    NaN
100229  PSE  State of Palestine    NaN

[198 rows x 3 columns]
```

Access all data for a specific party:

[3]:
```
reader.query(party_code="AFG")
```

[3]:
```
    party                    category      classification  \
0     AFG                  1.  Energy  Total for category
1     AFG                  1.  Energy  Total for category
2     AFG                  1.  Energy  Total for category
3     AFG                  1.  Energy  Total for category
4     AFG                  1.  Energy  Total for category
..    ...                         ...                 ...
376   AFG  unknown category nr. 10504  Total for category
377   AFG  unknown category nr. 10504  Total for category
378   AFG  unknown category nr. 10504  Total for category
379   AFG  unknown category nr. 10504  Total for category
380   AFG  unknown category nr. 10504  Total for category

                    measure             gas               unit  year  \
0    Net emissions/removals  Aggregate GHGs  Gg CO2 equivalent  2005
```

(continues on next page)

(continued from previous page)

```
1    Net emissions/removals  Aggregate GHGs  Gg CO2 equivalent  2013
2    Net emissions/removals             CH4                 Gg  2005
3    Net emissions/removals             CH4                 Gg  2013
4    Net emissions/removals              CO                 Gg  2005
..                      ...             ...                ...   ...
376        Total population          No gas          thousands  2015
377        Total population          No gas          thousands  2016
378        Total population          No gas          thousands  2017
379        Total population          No gas          thousands  2018
380        Total population          No gas          thousands  2019

     numberValue stringValue
0     3776.15946        None
1    10343.00000        None
2       35.06076        None
3        4.00000        None
4      865.20000        None
..           ...         ...
376  27101.36500        None
377  27657.14500        None
378  28224.32300        None
379  30075.01800        None
380  30725.56000        None

[381 rows x 9 columns]
```

Queries can be more specific, e.g. selecting only data about one gas:

[4]:
```
reader.query(party_code="DEU", gases=["N2O"])
```

[4]:
```
      party                                          category  \
0       DEU                                        1.  Energy
1       DEU                                        1.  Energy
2       DEU                                        1.  Energy
3       DEU                                        1.  Energy
4       DEU                                        1.  Energy
...     ...                                               ...
35440   DEU  Waste Incineration with Energy Recovery includ...
35441   DEU  Waste Incineration with Energy Recovery includ...
35442   DEU  Waste Incineration with Energy Recovery includ...
35443   DEU  Waste Incineration with Energy Recovery includ...
35444   DEU  Waste Incineration with Energy Recovery includ...

           classification                      measure  gas     unit  \
0      Total for category  Emission factor information  N2O  no unit
1      Total for category  Emission factor information  N2O  no unit
2      Total for category  Emission factor information  N2O  no unit
3      Total for category  Emission factor information  N2O  no unit
4      Total for category  Emission factor information  N2O  no unit
...                   ...                          ...  ...      ...
35440  Total for category       Net emissions/removals  N2O       kt
35441  Total for category       Net emissions/removals  N2O       kt
35442  Total for category       Net emissions/removals  N2O       kt
35443  Total for category       Net emissions/removals  N2O       kt
35444  Total for category       Net emissions/removals  N2O       kt

            year  numberValue stringValue
```

(continues on next page)

(continued from previous page)

```
0           1990          NaN      CS,D,M
1           1991          NaN      CS,D,M
2           1992          NaN      CS,D,M
3           1993          NaN      CS,D,M
4           1994          NaN      CS,D,M
...          ...          ...         ...
35440       2016          NaN          NO
35441       2017          NaN          NO
35442       2018          NaN          NO
35443       2019          NaN          NO
35444  Base year          NaN          NO

[35445 rows x 9 columns]
```

Queries can be made much more specific, if you want to. Check the API docs section (next section), or the docstrings for details.

CHAPTER

FOUR

API

4.1 unfccc_di_api

The API provided by the UNFCCC distinguishes between parties listed in Annex I and the other parties, likely because the reporting requirements for Annex I parties and non-Annex I parties differ substantially. This library provides a wrapper class *UNFCCCApiReader* which unifies both APIs so that you don't have to worry about the status of a particular party. However, if you want to filter for specific variables and only query a subset of the data, you have to use the individual API objects for Annex I and non-Annex I parties, which are available at *UNFCCCApiReader.annex_one_reader* and *UNFCCCApiReader.non_annex_one_reader*, respectively.

class unfccc_di_api.**UNFCCCApiReader**(*, *base_url: str = 'https://di.unfccc.int/api/'*)

Provides simplified unified access to the Flexible Query API of the UNFCCC data access for all parties.

Essentially encapsulates https://di.unfccc.int/flex_non_annex1 and https://di.unfccc.int/flex_annex1 .

parties

All parties, with their ID, code, and full name.

Type pandas.DataFrame

gases

The available gases and their IDs.

Type pandas.DataFrame

annex_one_reader

The API reader object for Annex I parties.

Type *UNFCCCSingleCategoryApiReader*

non_annex_one_reader

The API reader object for non-Annex I parties.

Type *UNFCCCSingleCategoryApiReader*

query(*, *party_code: str*, *gases: Optional[Sequence[str]] = None*, *progress: bool = False*, *normalize_gas_names: bool = True*) → pandas.core.frame.DataFrame

Query the UNFCCC for data.

Parameters

- **party_code** (*str*) – ISO code of a party for which to query. For possible values, see *parties*.
- **gases** (*list of str, optional*) – Limit the query to these gases. For possible values, see *gases*. Accepts subscripts ("N_2O") as well as ASCII-strings ("N2O"). Default: query for all gases.
- **progress** (*bool*) – Display a progress bar. Requires the `tqdm` library. Default: false.
- **normalize_gas_names** (*bool, optional*) – If `True`, return gases as ASCII strings ("N2O"). Else, return native UNFCCC notation ("N_2O"). Default: true.

Returns

Return type pandas.DataFrame

Notes

If you need more fine-grained control over which variables to query for, including restricting the query to specific measures, categories, or classifications or to query for multiple parties at once, please see the corresponding methods `UNFCCCApiReader.annex_one_reader.query()` and `UNFCCCApiReader.non_annex_one_reader.query()`.

class unfccc_di_api.**UNFCCCSingleCategoryApiReader**(*, *party_category: str*, *base_url: str = 'https://di.unfccc.int/api/'*)

Provides access to the Flexible Query API of the UNFCCC data access for a single category, either annexOne or nonAnnexOne.

Use this class if you want to do fine-grained queries for specific measures, categories, years, or classifications.

Essentially encapsulates https://di.unfccc.int/flex_non_annex1 or https://di.unfccc.int/flex_annex1 .

parties
All parties in this category, with their ID, code, and full name.

Type pandas.DataFrame

years
All years for which data is available, mapping the ID to the year.

Type pandas.DataFrame

category_tree
The available categories and their relationships. Use *show_category_hierarchy()* for displaying the category tree.

Type treelib.Tree

classifications
All classifications and their IDs.

Type pandas.DataFrame

measure_tree
The available measures and their relationsips. Use *show_measure_hierarchy()* for displaying the measure tree.

Type treelib.Tree

gases
The available gases and their IDs.

Type pandas.DataFrame

units
The available units and their IDs.

Type pandas.DataFrame

conversion_factors
Conversion factors between units for the specified gases.

Type pandas.DataFrame

variables
The available variables with the corresponding category, classification, measure, gas, and unit.

Type pandas.DataFrame

query(*, *party_codes: Sequence[str]*, *category_ids: Optional[Sequence[int]] = None*, *classifications: Optional[Sequence[str]] = None*, *measure_ids: Optional[Sequence[int]] = None*, *gases: Optional[Sequence[str]] = None*, *batch_size: int = 1000*, *progress: bool = False*, *normalize_gas_names: bool = True*) → pandas.core.frame.DataFrame

Query the UNFCCC for data.

Parameters

- **party_codes** (*list of str*) – List of ISO codes of parties for which to query. For possible values, see *parties*.
- **category_ids** (*list of int, optional*) – List of category IDs to query. For possible values, see *show_category_hierarchy()*. Default: query for all categories.
- **classifications** (*list of str, optional*) – List of classifications to query. For possible values, see *classifications*. Default: query for all classifications.
- **measure_ids** (*list of int, optional*) – List of measure IDs to query. For possible values, see *show_measure_hierarchy()*. Default: query for all measures.
- **gases** (*list of str, optional*) – Limit the query to these gases. For possible values, see *gases*. Accepts subscripts ("N_2O") as well as ASCII-strings ("N2O"). Default: query for all gases.
- **batch_size** (*int, optional*) – Number of variables to query in a single API query in the same batch to avoid internal server errors. Larger queries are split automatically. The default is 1000, which seems to work fine.
- **progress** (*bool*) – Display a progress bar. Requires the `tqdm` library. Default: false.
- **normalize_gas_names** (*bool, optional*) – If `True`, return gases as ASCII strings ("N2O"). Else, return native UNFCCC notation ("N_2O"). Default: true.

Returns

Return type pandas.DataFrame

Notes

Further documentation about the meaning of parties, categories, classifications, measures and gases is available at the UNFCCC documentation.

show_category_hierarchy() → None

Print the hierarchy of categories and their IDs.

show_measure_hierarchy() → None

Print the hierarchy of measures and their IDs.

4.2 unfccc_di_api.tests

Unit test package for unfccc_di_api.

Code coverage metrics:

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/pik-primap/unfccc_di_api/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Write Documentation

UNFCCC DI API could always use more documentation, whether as part of the official UNFCCC DI API docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.4 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/pik-primap/unfccc_di_api/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *unfccc_di_api* for local development.

1. Fork the *unfccc_di_api* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/unfccc_di_api.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ cd unfccc_di_api/
$ make virtual-environment
$ make install-pre-commit
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass our tests and automatically format everything according to our rules:

```
$ make lint

Often, the linters can fix errors themselves, so if you get failures, run
``make lint`` again to see if any errors need human intervention.
```

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6, 3.7, and 3.8.

5.4 Deploying

A reminder for the maintainers on how to deploy.

1. Commit all your changes.
2. Replace the unreleased entry in CHANGELOG.rst with your target version number.
3. Run `tbump X.Y.Z`.
4. Go to github and make a release from the tag. Use "Version x.y.z" as the release title, and the changelog entries as the release description. Creating the github release will automatically trigger a release on zenodo.
5. Run `make update-citation` to update the citation information in the README.
6. Upload the release to pyPI: `make release`
7. To prepare for future development, add a new "unreleased" section to CHANGELOG.rst, and commit the result.

CHAPTER

SIX

CREDITS

6.1 Developers

- Mika Pflüger mailto:mika.pflueger@pik-potsdam.de

6.2 Contributors

- Florian Dierickx floriandierickx.github.io
- Daniel Huppmann https://twitter.com/daniel_huppmann

6.3 Libraries

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

SEVEN

CHANGELOG

7.1 2.0.1 (2021-04-23)

- Change build system.

7.2 2.0.0 (2021-02-09)

- Accept ASCII format for `gases` when querying data and return gases & units normalized to ASCII (optional), thanks to Daniel Huppmann. Note that gases and units are normalized to ASCII by default, if you need the old behaviour for compatibility reasons, pass `normalize_gas_names=False` to your `query()` calls.

7.3 1.1.1 (2021-02-08)

- Include ipython notebooks and CHANGELOG in release tarballs.

7.4 1.1.0 (2021-01-25)

- Add a useful error message when querying for unknown parties, thanks to Daniel Huppmann.

7.5 1.0.0 (2021-01-22)

- Add continuous integration using GitHub actions.
- Add tests.
- Add usage documentation in notebook format.
- Documentation fixes.

7.6 0.1.0 (2021-01-22)

- First release on PyPI.
- Convert API wrapper into standalone Python package.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

u

unfccc_di_api, 9
unfccc_di_api.tests, 11

INDEX

A

annex_one_reader (*unfccc_di_api.UNFCCCApiReader attribute*), 9

C

category_tree (*unfccc_di_api.UNFCCCSingleCategoryApiReader attribute*), 10
classifications (*unfccc_di_api.UNFCCCSingleCategoryApiReader attribute*), 10
conversion_factors (*unfccc_di_api.UNFCCCSingleCategoryApiReader attribute*), 10

G

gases (*unfccc_di_api.UNFCCCApiReader attribute*), 9
gases (*unfccc_di_api.UNFCCCSingleCategoryApiReader attribute*), 10

M

measure_tree (*unfccc_di_api.UNFCCCSingleCategoryApiReader attribute*), 10
module
    unfccc_di_api, 9
    unfccc_di_api.tests, 11

N

non_annex_one_reader (*unfccc_di_api.UNFCCCApiReader attribute*), 9

P

parties (*unfccc_di_api.UNFCCCApiReader attribute*), 9
parties (*unfccc_di_api.UNFCCCSingleCategoryApiReader attribute*), 10

Q

query() (*unfccc_di_api.UNFCCCApiReader method*), 9
query() (*unfccc_di_api.UNFCCCSingleCategoryApiReader method*), 10

S

show_category_hierarchy() (*unfccc_di_api.UNFCCCSingleCategoryApiReader method*), 11
show_measure_hierarchy() (*unfccc_di_api.UNFCCCSingleCategoryApiReader method*), 11

U

unfccc_di_api
    module, 9
unfccc_di_api.tests
    module, 11
UNFCCCApiReader (*class in unfccc_di_api*), 9
UNFCCCSingleCategoryApiReader (*class in unfccc_di_api*), 10
units (*unfccc_di_api.UNFCCCSingleCategoryApiReader attribute*), 10

V

variables (*unfccc_di_api.UNFCCCSingleCategoryApiReader attribute*), 10

Y

years (*unfccc_di_api.UNFCCCSingleCategoryApiReader attribute*), 10