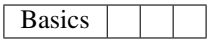
cmip6-data-citation-generator Documentation

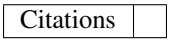
Release 0.3.1+2.gf741841

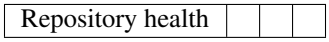
Zebedee Nicholls, Martina Stockhause, Paul Durack

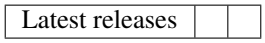
Dec 21, 2018

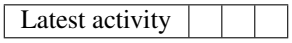
Documentation











The CMIP6 Data Citation Generator is free software under a BSD 2-Clause License, see LICENSE. If you make any use of the CMIP6 Data Citation Generator, please cite the relevant Zenodo release.

CHAPTER 1

Installation

The CMIP6 Data Citation Generator can be installed with pip.

```
# if you're using a virtual environment, make sure you're in it
pip install cmip6-data-citation-generator
```

CHAPTER 2

Usage

2.1 Generating json files

Having cloned this repository, an example, marked up, yaml file is given in `tests/test_data/valid_input.yaml`. Assuming that your working directory is the root of this repository, json files can then be generated as shown below. Running this command will produce output in the path `./example-outputs` along with output like the block below (note: any warning about Iris not being installed can be safely ignored). Each example json file is based off the template file `tests/test_data/valid_input.yaml` but fills in the missing text with information taken from the filepath of each data file.

```
# check current working directory
$ pwd
.../CMIP6-json-data-citation-generator
$ generate-cmip6-citation-files tests/test_data/input4MIPs_like tests/test_data/valid_
↪input.yaml ./example-outputs --drs CMIP6input4MIPs --regexp ".*\.nc" --keep
./example-outputs does not exist, making it now

Writing citation file for input4MIPs.CMIP6.AerChemMIP.UoM.UoM-AIM-ssp370-lowNTCF-1-2-
↪0 to ./example-outputs/input4MIPs.CMIP6.AerChemMIP.UoM.UoM-AIM-ssp370-lowNTCF-1-2-0.
↪json
Writing citation file for input4MIPs.CMIP6.CMIP.UoM.UoM-CMIP-1-2-0 to ./example-
↪outputs/input4MIPs.CMIP6.CMIP.UoM.UoM-CMIP-1-2-0.json
Writing citation file for input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-AIM-ssp370-1-2-0 to ./
↪example-outputs/input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-AIM-ssp370-1-2-0.json
Writing citation file for input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-GCAM4-ssp434-1-2-0 to .
↪/example-outputs/input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-GCAM4-ssp434-1-2-0.json
Writing citation file for input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-GCAM4-ssp460-1-2-0 to .
↪/example-outputs/input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-GCAM4-ssp460-1-2-0.json
Writing citation file for input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-IMAGE-ssp119-1-2-0 to .
↪/example-outputs/input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-IMAGE-ssp119-1-2-0.json
Writing citation file for input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-IMAGE-ssp126-1-2-0 to .
↪/example-outputs/input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-IMAGE-ssp126-1-2-0.json
Writing citation file for input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-MESSAGE-GLOBIOM-ssp245-
↪1-2-0 to ./example-outputs/input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-MESSAGE-GLOBIOM-
↪ssp245-1-2-0.json
```

(continues on next page)

(continued from previous page)

```
Writing citation file for input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-REMIND-MAGPIE-ssp534-
↪over-1-2-0 to ./example-outputs/input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-REMIND-MAGPIE-
↪ssp534-over-1-2-0.json
Writing citation file for input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-REMIND-MAGPIE-ssp585-1-
↪2-0 to ./example-outputs/input4MIPs.CMIP6.ScenarioMIP.UoM.UoM-REMIND-MAGPIE-ssp585-
↪1-2-0.json
```

Further help can be accessed with

```
$ generate-cmip6-citation-files -h
```

2.2 Uploading json files

json files can be upload to the CMIP6 data citation server using the command line.

To make this run, two vital steps must be taken:

1. Produce valid json files to upload (see *Generating json files*)
2. Meet the preconditions specified in Section 2.1 of the CMIP6 Citation Userguide

When installed, the upload client can be run with

```
$ upload-cmip6-citation-files input
```

where `input` is either a single file or a folder of files to upload. Further help can be accessed with

```
$ upload-cmip6-citation-files -h
```

CHAPTER 3

Development

If you're interested in contributing to the CMIP6 Data Citation Generator, we'd love to have you on board! This section of the docs details how to get setup to contribute and how best to communicate.

3.1 Contributing

All contributions are welcome, some possible suggestions include:

- tutorials (or support questions which, once solved, result in a new tutorial :D)

- blog posts
- improving the documentation
- bug reports
- feature requests
- pull requests

Please report issues or discuss feature requests in the CMIP6 Data Citation Generator issue tracker. If your issue is a feature request or a bug, please use the templates available, otherwise, simply open a normal issue :)

As a contributor, please follow a couple of conventions:

- Create issues in the CMIP6 Data Citation Generator issue tracker for changes and enhancements, this ensures that everyone in the community has a chance to comment
- Be welcoming to newcomers and encourage diverse new contributors from all backgrounds: see the Python Community Code of Conduct

3.2 Getting setup

To get setup as a developer, we recommend the following steps (if any of these tools are unfamiliar, please see the resources we recommend in *Development tools*):

1. Install make
2. Run `make venv`, if that fails the commands are
   (a) Create a Python virtual environment, `python3 -m venv venv`
   (b) Activate your virtual environment, `source venv/bin/activate` (on bash, other shells may be different)
   (c) Upgrade pip `pip install --upgrade pip`
   (d) Install an editable version of the CMIP6 Data Citation Generator along with development dependencies, `pip install -e .[test,docs,deploy]`
3. Make sure the tests pass by running `make test`, if that files the commands are
   (a) Run the unit and integration tests `./venv/bin/pytest --cov -rfsxEX --cov-report term-missing`

3.2.1 Getting help

Whilst developing, unexpected things can go wrong (that's why it's called 'developing', if we knew what we were doing, it would already be 'developed'). Normally, the fastest way to solve an issue is to contact us via the issue tracker. The other option is to debug yourself. For this purpose, we provide a list of the tools we use during our development as starting points for your search to find what has gone wrong.

Development tools

This list of development tools is what we rely on to develop the CMIP6 Data Citation Generator reliably and reproducibly. It gives you a few starting points in case things do go inexplicably wrong and you want to work out why. We include links with each of these tools to starting points that we think are useful, in case you want to learn more.

- Git

- Make
- **Tests**
  - we use a blend of pytest and the inbuilt Python testing capabilities for our tests so checkout what we've already done in `tests` to get a feel for how it works
- **Continuous integration (CI)**
  - we use Travis CI for our CI but there are a number of good providers
- Sphinx

Other tools

We also use some other tools which aren't necessarily the most familiar. Here we provide a list of these along with useful resources.

- **Regular expressions**
  - we use regex101.com to help us write and check our regular expressions, make sure the language is set to Python to make your life easy!

3.3 Formatting

To help us focus on what the code does, not how it looks, we use a couple of automatic formatting tools. These automatically format the code for us and tell use where the errors are. To use them, after setting yourself up (see *Getting setup*), simply run `make black` and `make flake8`. Note that `make black` can only be run if you have committed all your work i.e. your working directory is 'clean'. This restriction is made to ensure that you don't format code without being able to undo it, just in case something goes wrong.

3.4 Buiding the docs

After setting yourself up (see *Getting setup*), building the docs is as simple as running `make docs` (note, run `make -B docs` to force the docs to rebuild and ignore make when it says '... index.html is up to date'). This will build the docs for you. You can preview them by opening `docs/_build/html/index.html` in a browser.

For documentation we use Sphinx. To get ourselves started with Sphinx, we started with this example then used Sphinx's getting started guide.

3.4.1 Gotchas

To get Sphinx to generate pdfs (rarely worth the hassle as they're automatically built on Read the Docs anyway), you require Latexmk. On a Mac this can be installed with `sudo tlmgr install latexmk`. You will most likely also need to install some other packages (if you don't have the full distribution). You can check which package contains any missing files with `tlmgr search --global --file [filename]`. You can then install the packages with `sudo tlmgr install [package]`.

3.4.2 Docstring style

For our docstrings we use numpy style docstrings. For more information on these, here is the full guide and the quick reference we also use.

3.5 Releasing

The steps to release a new version of the CMIP6 Data Citation Generator are shown below. Please do all the steps below and all the steps for both release platforms.

1. Test installation with dependencies `make test-install`
2. Update `CHANGELOG.rst`:
   - add a header for the new version between `master` and the latest bullet point
   - this should leave the section underneath the master header empty
3. `git add .`
4. `git commit -m "Prepare for release of vX.Y.Z"`
5. `git push`
6. `git tag vX.Y.Z`
7. `git push --tags`

3.5.1 PyPI

1. `make publish-on-testpypi`
2. Go to test PyPI and check that the new release is as intended. If it isn't, stop and debug.
3. Test the install with `make test-testpypi-install` (this doesn't test all the imports as most required packages are not on test PyPI).
4. `make publish-on-pypi`
5. Go to the CMIP6 Data Citation Generator's PyPI and check that the new release is as intended.
6. Test the install with `make test-pypi-install` (a pip only install will throw warnings about Iris not being installed, that's fine).

3.5.2 Last steps

1. If you want to archive this version, follow the instructions here
2. Update any badges in `README.rst` that don't update automatically (note that the commits since badge only updates if you archive the version)
3. `git add .`
4. `git commit -m "Update README badges"`
5. `git push`

3.6 Why is there a `Makefile` in a pure Python repository?

Whilst it may not be standard practice, a `Makefile` is a simple way to automate general setup (environment setup in particular). Hence we have one here which basically acts as a notes file for how to do all those little jobs which we often forget e.g. setting up environments, running tests (and making sure we're in the right environment), building docs, setting up auxillary bits and pieces.

3.7 Why did we choose a BSD 2-Clause License?

We want to ensure that our code can be used and shared as easily as possible. Whilst we love transparency, we didn't want to **force** all future users to also comply with a stronger license such as AGPL. Hence the choice we made.

We recommend Morin et al. 2012 for more information for scientists about open-source software licenses.

CHAPTER 4

cmip6_data_citation_generator API

cmip6_data_citation_generator.**generate_jsons**(*input_dir*, *template_yaml*, *drs*, *output_dir*, *regexp='.*'*, *keep=True*)

Generate CMIP6 data citation json files

Parameters

- **input_dir** (*str*) – Directory to search for files.
- **template_yaml** (*str*) – Path to yaml file to use as a template for generating the json file.
- **drs** (*str*) – The data reference syntax used to save your data. Must be one of ["CMIP6input4MIPs", "CMIP6output"].
- **output_dir** (*str*) – The path in which to save the generated files.
- **regexp** (*str*) – Regular expression to use to filter the filepaths found in `input_dir`.
- **keep** (*bool*) – If True, generate jsons for the files in filepaths which match `regexp`. If False, do the opposite i.e. generate jsons for the files in filepaths which don't match `regexp`.

CHAPTER 5

cmip6_data_citation_generator.utils API

cmip6_data_citation_generator.utils.**deep_substitute**(*input_val*, *substitutions*)

Substitute strings in the input recursively

In the input, strings contained between carets e.g. '<name>', will be replaced with the corresponding value in `substitutions`. Before looking for a replacement, the carets are removed. For example, '<name>' will be replaced with `substitutions["name"]`.

Note that keys in any input value which is a dictionary will not be replaced, see the examples.

Parameters

- **input_val** (*str, float, int, list, dict, nested structures of the above*) – The object in which the substitutions should be made
- **substitutions** (*dict*) – The substitutions to make

Returns The input_val with all substitutions made

Return type type(input_val)

Raises `KeyError` – If no substitution can be found

Examples

```
>>> deep_substitute("<source_id>", {"source_id": "UoM"})
'UoM'
```

```
>>> deep_substitute(["<source_id>", "other string"], {"source_id": "UoM"})
['UoM', 'other string']
```

```
>>> deep_substitute([["<source_id>", "other string"], "<activity_id>"], {"source_
↪id": "UoM", "activity_id": "21st Century runs"})
[['UoM', 'other string'], '21st Century runs']
```

```
>>> deep_substitute([{"other string": "<source_id>"}, "<activity_id>"], {"source_
↪id": "UoM", "activity_id": "21st Century runs"})
[{'other string': 'UoM'}, '21st Century runs']
```

```
>>> # keys in input dictionaries are not substituted
>>> deep_substitute({"<source_id>": "<source_id>"}, {"source_id": "UoM"})
{'<source_id>': 'UoM'}
```

```
>>> # missing substitutions will raise ``KeyError``
>>> deep_substitute("<source_id>", {"activity_id": "21st Century runs"})
KeyError: "No substitution provided for ['<source_id>']"
```

CHAPTER 6

cmip6_data_citation_generator.io_dcg API

cmip6_data_citation_generator.io_dcg.**load_and_validate_yaml**(*yaml_to_read*, *schema=<class 'cmip6_data_citation_generator.validators.C*

Load yaml from file and validate using schema

Parameters

- **yaml_to_read** (*str*) – File to read the yaml from
- **schema** (marshmallow.Schema) – Schema to use to validate the loaded yaml

Returns Loaded dictionary which has been validated by schema

Return type dict

cmip6_data_citation_generator.io_dcg.**validate_and_return_raw_dict**(*raw_dict*, *schema=<class 'cmip6_data_citation_generator.vali*

Return dictionary validated using the CitationSchema

Parameters

- **raw_dict** (*dict*) – Raw dictionary to load
- **schema** (marshmallow.Schema) – Schema to use to validate the dictionary

Returns Loaded dictionary which has been validated by schema

Return type dict

cmip6_data_citation_generator.io_dcg.**write_json**(*json_dict*, *path*)

Write json file from input dictionary to path

Parameters

- **json_dict** (*dict*) – Dictionary containing the json to write
- **path** (*str*) – Path to write to

CHAPTER 7

Changelog

7.1 master

7.2 v0.3.1

- Nothing new, just updating to get history and Zenodo citation right

7.3 v0.3.0

- (#25) Add command line interface for generation as well as uploading code

7.4 v0.2.0

- (#23) Fix code coverage, automatically add subject and improve example yaml
- (#22) Added test of special character support

7.5 v0.1.2

- Nothing new, just updating to get Zenodo citation right

7.6 v0.1.1

- Fix Python version id for PyPI

7.7 v0.1.0

- (#18) Added basic citation generator setup

CHAPTER 8

Other Resources

The complete CMIP6 Data Citation User Guide can be found at https://cera-www.dkrz.de/docs/pdf/CMIP6_Citation_Userguide.pdf. The last page in particular is very helpful. It documents and explains a number of error messages which can arise when uploading.

CHAPTER 9

Index

- genindex
- modindex
- search

Python Module Index

C

Index