# copernicus-wps-demo Documentation

*Release 0.3.0*

**Carsten Ehbrecht**

**Jul 05, 2018**

# Contents:

Demo with WPS processes for Copernicus.

- Free software: Apache Software License 2.0
- Documentation: https://copernicus-wps-demo.readthedocs.io.

CHAPTER 1

# Credits

This package was created with Cookiecutter and the bird-house/cookiecutter-birdhouse project template.

## 1.1 Installation

### 1.1.1 Install from Anaconda

### 1.1.2 Install from GitHub

Check out code from the copernicus-wps-demo GitHub repo and start the installation:

```
$ git clone https://github.com/cp4cds/copernicus-wps-demo.git
$ cd copernicus-wps-demo
$ conda env create -f environment.yml
$ source activate copernicus
$ python setup.py develop
```

#### ... or do it the lazy way

The previous installation instructions assume you have Anaconda installed. We provide also a `Makefile` to run this installation without additional steps:

```
$ git clone https://github.com/cp4cds/copernicus-wps-demo.git
$ cd copernicus-wps-demo
$ make clean    # cleans up a previous Conda environment
$ make install  # installs Conda if necessary and runs the above installation steps
```

### 1.1.3 Start copernicus-wps-demo PyWPS service

After successful installation you can start the service using the `copernicus` command-line.

```
$ copernicus --help # show help
$ copernicus start  # start service with default configuration

OR

$ copernicus start --daemon # start service as daemon
loading configuration
forked process id: 42
```

The deployed WPS service is by default available on:

http://localhost:5000/wps?service=WPS&version=1.0.0&request=GetCapabilities.

**Note:** Remember the process ID (PID) so you can stop the service with `kill PID`.

Check the log files for errors:

```
$ tail -f  pywps.log
```

#### ... or do it the lazy way

You can also use the `Makefile` to start and stop the service:

```
$ make start
$ make status
$ tail -f pywps.log
$ make stop
```

### 1.1.4 Run copernicus-wps-demo as Docker container

You can also run copernicus-wps-demo as a Docker container, see the Tutorial.

### 1.1.5 Use Ansible to deploy copernicus-wps-demo on your System

Use the Ansible playbook for PyWPS to deploy copernicus-wps-demo on your system. Follow the example for copernicus-wps-demo given in the playbook.

### 1.1.6 Building the docs

First install dependencies for the documentation:

```
$ make bootstrap_dev
$ make docs
```

## 1.2 Configuration

### 1.2.1 Command-line options

You can overwrite the default PyWPS configuration by using command-line options. See the copernicus-wps-demo help which options are available:

```
$ copernicus start --help
--hostname HOSTNAME        hostname in PyWPS configuration.
--port PORT                port in PyWPS configuration.
```

Start service with different hostname and port:

```
$ copernicus start --hostname localhost --port 5001
```

### 1.2.2 Use a custom configuration file

You can overwrite the default PyWPS configuration by providing your own PyWPS configuration file (just modifiy the options you want to change). Use one of the existing `sample-*.cfg` files as example and copy them to `etc/custom.cfg`.

For example change the hostname (*demo.org*) and logging level:

```
$ cd copernicus-wps-demo
$ vim etc/custom.cfg
$ cat etc/custom.cfg
[server]
url = http://demo.org:5000/wps
outputurl = http://demo.org:5000/outputs

[logging]
level = DEBUG
```

Start the service with your custom configuration:

```
# start the service with this configuration
$ copernicus start -c etc/custom.cfg
```

## 1.3 Processes

- *Sleep*
- *Wordcounter*
- *InOut*
- *MyDiag*
- *Perfmetrics*
- *PyDemo*
- *RainFarm*

- *RMSE*

### 1.3.1 Sleep

**class** copernicus.processes.wps_sleep.**Sleep**

**sleep** Sleep Process (v1.0)

Testing a long running process, in the sleep.This process will sleep for a given delay or 10 seconds if not a valid value.

**Parameters** **delay** (*float*) – Delay between every update

**Returns** **sleep_output** – Sleep Output

**Return type** string

#### References

- PyWPS Demo

### 1.3.2 Wordcounter

**class** copernicus.processes.wps_wordcounter.**WordCounter**

**wordcounter** Word Counter (v1.0)

Counts words in a given text.

**Parameters** **text** (*text/plain*) – URL pointing to a text document, for example "Alice in Wonderland": http://www.gutenberg.org/cache/epub/19033/pg19033.txt

**Returns** **output** – Word counter result

**Return type** *application/json*

#### References

- User Guide
- Free eBooks at Gutenberg
- Example: Alice in Wonderland

Counts ocurrences of all words in a document.

### 1.3.3 InOut

**class** copernicus.processes.wps_inout.**InOut**

**inout** In and Out (v1.0)

Testing all WPS input and output parameters.

**Parameters**

- **string** (*string*) – Enter a simple string.

- **int** (`{'1', '2', '3', '5', '7', '11'}`) – Choose an integer number from allowed values.
- **float** (`float`) – Enter a float number.
- **boolean** (`boolean`) – Make your choice :)
- **time** (`time`) – Enter a time like 12:00:00
- **date** (`date`) – Enter a date like 2012-05-01
- **datetime** (`dateTime`) – Enter a datetime like 2016-09-02T12:00:00Z
- **string_choice** (`{'rock', 'paper', 'scissor'}`) – Choose one item form list.
- **string_multiple_choice** (`{'sitting duck', 'flying goose', 'happy pinguin', 'gentle albatros'}`) – Choose one or two items from list.
- **text** (`text/plain`, optional) – Enter a URL pointing to a text document (optional) (Info)
- **dataset** (`application/x-netcdf`, optional) – Enter a URL pointing to a NetCDF file (optional) (NetCDF Format)

**Returns**

- **string** (*string*) – String
- **int** (*integer*) – Integer
- **float** (*float*) – Float
- **boolean** (*boolean*) – Boolean
- **time** (*time*) – Time
- **date** (*date*) – Date
- **datetime** (*dateTime*) – DateTime
- **string_choice** (*string*) – String Choice
- **string_multiple_choice** (*string*) – String Multiple Choice
- **text** (`text/plain`) – Copy of input text file.
- **dataset** (`application/x-netcdf`, `text/plain`) – Copy of input netcdf file.
- **bbox** (*[epsg:4326]*) – Bounding Box

### References

- Birdhouse
- User Guide

TODO: add literal input with value range[(0,100)] . . . see pywps doc

### 1.3.4 MyDiag

**class** copernicus.processes.wps_mydiag.**MyDiag**

**mydiag** Simple plot (v2.0.0)

Generates a plot for temperature using ESMValTool. It is a diagnostic used in the ESMValTool tutoriaal doc/toy-diagnostic-tutorial.pdf. The default run uses the following CMIP5 data: project=CMIP5, experiment=historical, ensemble=r1i1p1, variable=ta, model=MPI-ESM-LR, time_frequency=mon

**Parameters**

- **model** (*{'MPI-ESM-LR', 'MPI-ESM-MR'}*) – Choose a model like MPI-ESM-LR.
- **experiment** (*{'historical', 'rcp26', 'rcp45', 'rcp85'}*) – Choose an experiment like historical.
- **ensemble** (*{'r1i1p1', 'r2i1p1', 'r3i1p1'}*) – Choose an ensemble like r1i1p1.
- **start_year** (*integer*) – Start year of model data.
- **end_year** (*integer*) – End year of model data.

**Returns**

- **namelist** (*text/plain*) – ESMValTool namelist used for processing.
- **log** (*text/plain*) – Log File of ESMValTool processing.
- **output** (*application/pdf*) – Generated output plot of ESMValTool processing.

#### References

- ESMValTool
- ESGF Testdata

### 1.3.5 Perfmetrics

**class** copernicus.processes.wps_perfmetrics.**Perfmetrics**

**perfmetrics** Performance metrics (v2.0.0)

Creates a performance metrics report comparing models using ESMValTool. The goal is to create a standard namelist for the calculation of performance metrics to quantify the ability of the models to reproduce the climatological mean annual cycle for selected Essential Climate Variables (ECVs) plus some additional corresponding diagnostics and plots to better understand and interpret the results. The namelist can be used to calculate performance metrics at different vertical levels (e.g., 5, 30, 200, 850 hPa as in Gleckler et al., 2008) and in four regions (global, tropics 20N-20S, northern extratropics 20-90N, southern extratropics 20-90S). As an additional reference, we consider the Righi et al. (2015) paper.

**Parameters**

- **model** (*{'MPI-ESM-LR', 'MPI-ESM-MR'}*) – Choose a model like MPI-ESM-LR.
- **experiment** (*{'historical', 'rcp26', 'rcp45', 'rcp85'}*) – Choose an experiment like historical.
- **ensemble** (*{'r1i1p1', 'r2i1p1', 'r3i1p1'}*) – Choose an ensemble like r1i1p1.
- **start_year** (*integer*) – Start year of model data.

- **end_year** (*integer*) – End year of model data.

**Returns**

- **namelist** (*text/plain*) – ESMValTool namelist used for processing.
- **log** (*text/plain*) – Log File of ESMValTool processing.
- **output** (*application/pdf*) – Generated output plot of ESMValTool processing.

#### References

- ESMValTool
- Documentation
- Media
- Diagnostic Description
- Diagnostic Metadata

### 1.3.6 PyDemo

**class** copernicus.processes.wps_pydemo.**PyDemo**

**py_demo** Python Demo (v2.0.0)

Generates a plot for temperature using ESMValTool. The default run uses the following CMIP5 data: project=CMIP5, experiment=historical, ensemble=r1i1p1, variable=ta, model=MPI-ESM-LR, time_frequency=mon

**Parameters**

- **model** (*{'MPI-ESM-LR', 'MPI-ESM-MR'}*) – Choose a model like MPI-ESM-LR.
- **experiment** (*{'historical', 'rcp26', 'rcp45', 'rcp85'}*) – Choose an experiment like historical.
- **ensemble** (*{'r1i1p1', 'r2i1p1', 'r3i1p1'}*) – Choose an ensemble like r1i1p1.
- **start_year** (*integer*) – Start year of model data.
- **end_year** (*integer*) – End year of model data.

**Returns**

- **namelist** (*text/plain*) – ESMValTool namelist used for processing.
- **log** (*text/plain*) – Log File of ESMValTool processing.
- **output** (*image/png*, *application/pdf*) – Generated output plot of ESMValTool processing.

#### References

- ESMValTool
- ESGF Testdata

### 1.3.7 RainFarm

**class** copernicus.processes.wps_rainfarm.**RainFarm**

**rainfarm** RainFARM stochastic downscaling (v2.0.0)

Tool to perform stochastic precipitation downscaling, generating an ensemble of fine-scale precipitation fields from information simulated by climate models at regional scale.

**Parameters**

- **model** (*{'ACCESS1-0'}*) – Choose a model like MPI-ESM-LR.
- **experiment** (*{'historical'}*) – Choose an experiment like historical.
- **start_year** (*integer*) – Start year of model data.
- **end_year** (*integer*) – End year of model data.
- **subset** (*string*) – Choose a geographical subset with a Bounding Box: 4,13,44,53
- **regridding** (*boolean*) – Flag for regridding.
- **slope** (*boolean*) – Flag for slope.
- **num_ens_members** (*integer*) – Choose a number of ensemble members.
- **num_subdivs** (*integer*) – Choose a number of subdivisions.

**Returns** **output** – Generated output plot of ESMValTool processing.

**Return type** *image/png*

#### References

- ESMValTool
- Documentation
- Media
- Diagnostic Description
- Diagnostic Metadata

### 1.3.8 RMSE

**class** copernicus.processes.wps_rmse.**RMSE**

**rmse** Modes of variability (v2.0.0)

Tool to compute the RMSE between the observed and modelled patterns of variability obtained through classification and their relative relative bias (percentage) in the frequency of occurrence and the persistence of each mode.

**Parameters**

- **region** (*{'Arctic'}*) – Choose a region like Arctic.
- **model** (*{'NASA'}*) – Choose a model like NASA.
- **variable** (*{'sic'}*) – Choose a variable like sic.
- **ncenters** (*integer*) – Choose a number of centers.
- **cluster_method** (*{'kmeans'}*) – Choose a cluster method.

- **eofs** (*boolean*) – Choose EOFS.
- **detrend** (*integer*) – Choose a detrend.
- **experiment** (*string*) – Choose an experiment.

**Returns** **output** – Generated output plot of ESMValTool processing.

**Return type** *image/png*

#### References

- ESMValTool
- Documentation
- Media
- Diagnostic Description
- Diagnostic Metadata

## 1.4 Changes

### 1.4.1 0.3.0 (2018-06-22)

- Generated from cookiecutter template (#13).
- Skipped buildout just relying on conda and werkzeug.
- Use sphinx documentation (#12).
- Added "fake" processes rainfarm and rmse from MAGIC demo (#16, #22).
- Using static diagnostics description from MAGIC demo (#20).

### 1.4.2 0.2.2 (2018-06-18)

- Added WPS client examples (#7).
- Added WPS client examples with x509 certificate (#10).

### 1.4.3 0.2.1 (2018-02-08)

- fixed ncl installation (#3).
- added demo service using werkzeug.

### 1.4.4 0.2.0 (2018-02-06)

- using ESMValTool 2.x
- diag list: mydiag and py_demo

### 1.4.5 0.1.1 (2018-01-29)

- cleaned up Dockerfile.
- added pscopg2 for postgres.
- updated pywps recipe 0.9.3.
- added workaround for broken ncl conda package.
- updated Readme with birdy examples.

### 1.4.6 0.1.0 (2017-06-06)

- added tutorial diagnostics from esmvaltool (MyDiag, Overview, timeseriesplot).
- timeseriesplot both added with a generic wps profile (opendap, netcdf) and with esgf search facets.
- added NCL coutour plot with opendap and netcdf input.

CHAPTER 2

# Indices and tables

- genindex
- modindex
- search

# Index