
Malleefowl Documentation

Release 0.6

Birdhouse

Dec 05, 2018

Contents

1	Installation	3
2	Configuration	5
3	Developer Guide	7
4	Using Docker	9
5	Tutorials	11
6	Sphinx AutoAPI Index	19
7	Indices and tables	31
	Python Module Index	33

Malleefowl (the bird) *Malleefowl are shy, wary, solitary birds that usually fly only to escape danger or reach a tree to roost in. Although very active, they are seldom seen [..] (Wikipedia).*

Malleefowl is a [Web Processing Service](#) with a collection of processes to access climate data (ESGF, Thredds Catalogs, ...).

Malleefowl is part of the [Birdhouse](#) project.

Contents:

CHAPTER 1

Installation

Check out code from the malleefowl github repo and start the installation:

```
$ git clone https://github.com/bird-house/malleefowl.git
$ cd malleefowl
$ make clean install
```

For other install options run `make help` and read the documentation of the [Makefile](#). All installation files are going by default into the folder `~/birdhouse`.

After successful installation you need to start the services:

```
$ make start    # starts supervisor services
$ make status   # show supervisor status
```

The depolyed WPS service is available at:

<http://localhost:8091/wps?service=WPS&version=1.0.0&request=GetCapabilities>.

Check the log files for errors:

```
$ tail -f ~/birdhouse/var/log/pywps/malleefowl.log
$ tail -f ~/birdhouse/var/log/supervisor/malleefowl.log
```


CHAPTER 2

Configuration

If you want to run on a different hostname or port then change the default values in `custom.cfg`:

```
$ cd malleefowl
$ vim custom.cfg
$ cat custom.cfg
[settings]
hostname = localhost
http-port = 8091
```

After any change to your `custom.cfg` you **need** to run `make update` again and restart the supervisor service:

```
$ make update    # or install
$ make restart
$ make status
```


- *Running unit tests*
- *Running WPS service in test environment*

3.1 Running unit tests

Run quick tests:

```
$ make test
```

Run all tests (slow, online):

```
$ make testall
```

Check pep8:

```
$ make pep8
```

3.2 Running WPS service in test environment

For development purposes you can run the WPS service without nginx and supervisor. Use the following instructions:

```
# get the source code
$ git clone https://github.com/bird-house/malleefowl.git
$ cd malleefowl

# create conda environment
```

(continues on next page)

(continued from previous page)

```
$ conda env create -f environment.yml

# activate conda environment
$ source activate malleefowl

# install malleefowl code into conda environment
$ python setup.py develop

# start the WPS service
$ malleefowl

# open your browser on the default service url
$ firefox http://localhost:5000/wps

# ... and service capabilities url
$ firefox http://localhost:5000/wps?service=WPS&request=GetCapabilities
```

The malleefowl service command-line has more options:

```
$ malleefowl -h
```

For example you can start the WPS with enabled debug logging mode:

```
$ malleefowl --debug
```

Or you can overwrite the default **PyWPS** configuration by providing your own PyWPS configuration file (just modify the options you want to change):

```
# edit your local pywps configuration file
$ cat mydev.cfg
[logging]
level = WARN
file = /tmp/mydev.log

# start the service with this configuration
$ malleefowl -c mydev.cfg
```

CHAPTER 4

Using Docker

To run Malleefowl Web Processing Service you can also use the [Docker](#) image:

```
$ docker run -i -d -p 9001:9001 -p 8000:8000 -p 8080:8080 --name=malleefowl birdhouse/  
↪malleefowl
```

Check the docker logs:

```
$ docker logs malleefowl
```

Show running docker containers:

```
$ docker ps
```

Open your browser and enter the url of the supervisor service:

<http://localhost:9001/>

Run a GetCapabilities WPS request:

<http://localhost:8080/wps?service=WPS&version=1.0.0&request=getcapabilities>

4.1 Using docker-compose

Start malleefowl with docker-compose (docker-compose version > 1.7):

```
$ docker-compose up
```

By default the WPS is available on port 8080: <http://localhost:8080/wps?service=WPS&version=1.0.0&request=GetCapabilities>.

You can change the ports and hostname with environment variables:

```
$ HOSTNAME=malleefowl HTTP_PORT=8091 SUPERVISOR_PORT=48091 docker-compose up
```

Now the WPS is available on port 8091: <http://malleefowl:8091/wps?service=WPS&version=1.0.0&request=GetCapabilities>.

5.1 Using the *download* Process

Go through this tutorial step by step.

- *Step 0: Install malleefowl with defaults*
- *Step 1: Install birdy*
- *Step 2: Check if birdy works*
- *Step 3: Run the download process*
- *Step 4: Install Phoenix*
- *Step 5: Login to Phoenix*
- *Step 6: Copy the twitcher access token in Phoenix*
- *Step 7: Access malleefowl behind the OWS proxy with access token*
- *Step 8: Get a ESGF certificate using Phoenix*
- *Step 9: Download a file from ESGF*

5.1.1 Step 0: Install malleefowl with defaults

```
# get the source code
$ git clone https://github.com/bird-house/malleefowl.git
$ cd malleefowl

# run the installation
$ make clean install
```

(continues on next page)

(continued from previous page)

```
# start the service
$ make start

# open the capabilities document
$ firefox http://localhost:8091/wps?service=WPS&request=GetCapabilities
```

5.1.2 Step 1: Install birdy

We are using birdy in the examples, a WPS command line client.

```
# install it via conda
$ conda install -c birdhouse birdhouse-birdy
```

5.1.3 Step 2: Check if birdy works

```
# point birdy to the malleefowl service url
$ export WPS_SERVICE=http://localhost:8091/wps
# show a list of available command (wps processes)
$ birdy -h
```

5.1.4 Step 3: Run the download process

Make sure birdy works and is pointing to malleefowl ... see above.

```
# show the description of the download process
$ birdy download -h

# download a netcdf file from a public thredds service
$ birdy download --resource \
    https://www.esrl.noaa.gov/psd/thredds/fileServer/Datasets/nccep.reanalysis2/
↪surface/mslp.1979.nc
```

5.1.5 Step 4: Install Phoenix

Phoenix is a web client for WPS and comes by default with an WPS security proxy (twitcher).

```
$ git clone https://github.com/bird-house/pyramid-phoenix.git
$ cd pyramid-phoenix
$ make clean install
$ make restart
```

5.1.6 Step 5: Login to Phoenix

```
# login ... by default admin password is "qwerty"
$ firefox https://localhost:8443/account/login
```


5.1.7 Step 6: Copy the twitcher access token in Phoenix

1. Go to your profile.
2. Choose the Twitcher access token tab.
3. Copy the access token.

5.1.8 Step 7: Access malleefowl behind the OWS proxy with access token

```
# configure wps service
$ export WPS_SERVICE=https://localhost:8443/ows/proxy/malleefowl

# check if it works
$ birdy -h

# run the download again ... you need the access token
$ birdy \
  --token 3d8c24eeebb143b3a199ba8a0e045f93 \
  download --resource \
    https://www.esrl.noaa.gov/psd/thredds/fileServer/Datasets/ncep.reanalysis2/
↪ surface/mslp.1979.nc
```

5.1.9 Step 8: Get a ESGF certificate using Phoenix

1. Go to your profile.
2. Choose the ESGF credentials tab.
3. Use the green button Update credentials.
4. Choose your ESGF provider, enter your account details and press Submit.

5.1.10 Step 9: Download a file from ESGF

Make sure birdy works and points to the proxy url of malleefowl ... see above.

Choose a file from the ESGF archive you would like to download and make sure you have download permissions.

You can choose the ESGF [search browser](#) in Phoenix or an [ESGF portal](#).

```
# try the download ... in this example with a CORDEX file.
# make sure your twitcher token and your ESGF cert are still valid.
$ birdy \
  --token 3d8c24eeebb143b3a199ba8a0e045f93 \
  download --resource \
    http://esgf1.dkrz.de/thredds/fileServer/cordex/cordex/output/EUR-44/MPI-CSC/MPI-M-
↪ MPI-ESM-LR/historical/r1ilp1/MPI-CSC-REMO2009/v1/mon/tas/v20150609/tas_EUR-44_MPI-M-
↪ MPI-ESM-LR_historical_r1ilp1_MPI-CSC-REMO2009_v1_mon_200101-200512.nc
```

5.2 Debugging the *download* Process

Go through this tutorial step by step.

- *Step 0: Install malleefowl in debug mode*
- *Step 1: Start the malleefowl demo service*
- *Step 2: Install birdy*
- *Step 3: Check if birdy works*
- *Step 4: Run the download process*
- *Step 5: Install Phoenix*
- *Step 6: Login to Phoenix*
- *Step 7: Register your WPS demo service*
- *Step 8: Copy the twitcher access token in Phoenix*
- *Step 9: Access demo service behind the OWS proxy with access token*
- *Step 10: Get an ESGF certificate using Phoenix*
- *Step 11: Download a file from ESGF*

5.2.1 Step 0: Install malleefowl in debug mode

```
# get the source code
$ git clone https://github.com/bird-house/malleefowl.git
$ cd malleefowl

# create conda env
$ conda env create

# activate malleefowl env
$ source activate malleefowl

# install malleefowl package in develop mode
$ python setup.py develop

# check if the demo service is available
$ malleefowl -h
```

5.2.2 Step 1: Start the malleefowl demo service

You might do this more often when debugging. Make sure you are in the malleefowl conda env.

```
# start service
$ malleefowl

# open the capabilities document
$ firefox http://localhost:5000/wps?service=WPS&request=GetCapabilities
```

The service is started in debug mode. See the [Werkzeug](#) documentation how to work with this.

You can stop the service with CTRL-C. The service is automatically restarted on source changes.

5.2.3 Step 2: Install birdy

We are using birdy in the examples, a WPS command line client.

```
# install it via conda
$ conda install -c birdhouse birdhouse-birdy
```

5.2.4 Step 3: Check if birdy works

```
# point birdy to the malleefowl service url
$ export WPS_SERVICE=http://localhost:5000/wps
# show a list of available command (wps processes)
$ birdy -h
```

5.2.5 Step 4: Run the download process

Make sure birdy works and is pointing to malleefowl ... see above.

```
# show the description of the download process
$ birdy download -h

# download a netcdf file from a public thredds service
$ birdy download --resource \
    https://www.esrl.noaa.gov/psd/thredds/fileServer/Datasets/ncep.reanalysis2/
↪ surface/mslp.1979.nc
```

5.2.6 Step 5: Install Phoenix

Phoenix is a web client for WPS and comes by default with an WPS security proxy (twitcher).

```
$ git clone https://github.com/bird-house/pyramid-phoenix.git
$ cd pyramid-phoenix
$ make clean install
$ make restart
```

5.2.7 Step 6: Login to Phoenix

```
# login ... by default admin password is "qwerty"
$ firefox https://localhost:8443/account/login
```

5.2.8 Step 7: Register your WPS demo service

Go to the registration page: <https://localhost:8443/services/register>

Register your service with the following parameters:

- Service URL: <http://localhost:5000/wps>
- Service Name: demo

5.2.9 Step 8: Copy the twitcher access token in Phoenix

1. Go to your profile.
2. Choose the Twitcher access token tab.
3. Copy the access token.

5.2.10 Step 9: Access demo service behind the OWS proxy with access token

```
# configure wps service
$ export WPS_SERVICE=https://localhost:8443/ows/proxy/demo

# check if it works
$ birdy -h

# run the download again ... you need the access token
$ birdy \
  --token 3d8c24eeebb143b3a199ba8a0e045f93 \
  download --resource \
    https://www.esrl.noaa.gov/psd/thredds/fileServer/Datasets/ncep.reanalysis2/
↪ surface/mslp.1979.nc
```

5.2.11 Step 10: Get an ESGF certificate using Phoenix

1. Go to your profile.
2. Choose the ESGF credentials tab.
3. Use the green button Update credentials.
4. Choose your ESGF provider, enter your account details and press Submit.

5.2.12 Step 11: Download a file from ESGF

Make sure birdy works and points to the proxy url of demo service ... see above.

Choose a file from the ESGF archive you would like to download and make sure you have download permissions.

You can choose the ESGF [search browser](#) in Phoenix or an [ESGF portal](#).

```
# try the download ... in this example with a CORDEX file.
# make sure your twitcher token and your ESGF cert are still valid.
$ birdy \
  --token 3d8c24eeebb143b3a199ba8a0e045f93 \
  download --resource \
    http://esgf1.dkrz.de/thredds/fileServer/cordex/cordex/output/EUR-44/MPI-CSC/MPI-M-
↪ MPI-ESM-LR/historical/r1ilp1/MPI-CSC-REMO2009/v1/mon/tas/v20150609/tas_EUR-44_MPI-M-
↪ MPI-ESM-LR_historical_r1ilp1_MPI-CSC-REMO2009_v1_mon_200101-200512.nc
```

You can also try this in WPS synchronous mode when your process is not long running:

```
$ birdy \
  --sync \
  --token 3d8c24eeebb143b3a199ba8a0e045f93 \
```

(continues on next page)

(continued from previous page)

```
download --resource \
http://esgfl.dkrz.de/thredds/fileServer/cordex/cordex/output/EUR-44/MPI-CSC/MPI-M-
↪MPI-ESM-LR/historical/r1ilp1/MPI-CSC-REMO2009/v1/mon/tas/v20150609/tas_EUR-44_MPI-M-
↪MPI-ESM-LR_historical_r1ilp1_MPI-CSC-REMO2009_v1_mon_200101-200512.nc
```

... and with debug option to see more log message:

```
$ birdy \
--sync \
--debug \
--token 3d8c24eeebb143b3a199ba8a0e045f93 \
download --resource \
http://esgfl.dkrz.de/thredds/fileServer/cordex/cordex/output/EUR-44/MPI-CSC/MPI-M-
↪MPI-ESM-LR/historical/r1ilp1/MPI-CSC-REMO2009/v1/mon/tas/v20150609/tas_EUR-44_MPI-M-
↪MPI-ESM-LR_historical_r1ilp1_MPI-CSC-REMO2009_v1_mon_200101-200512.nc
```


This page is the top-level of your generated API documentation. Below is a list of all items that are documented here.

6.1 malleefowl

6.1.1 Subpackages

`malleefowl.esgf`

Submodules

`malleefowl.esgf.logon`

This module is used to get esgf logon credentials. There are two choices:

- a proxy certificate from a myproxy server with an ESGF openid.
- OpenID login as used in browsers.

Some of the code is taken from esgf-pyclient: <https://github.com/ESGF/esgf-pyclient>

See also:

- open climate workbench: <https://github.com/apache/climate>
- MyProxyLogon: <https://github.com/cedadev/MyProxyClient>

Module Contents

`malleefowl.esgf.logon.logger`

```
malleefowl.esgf.logon.myproxy_logon_with_openid(openid, password=None, interactive=False, outdir=None)
```

Tries to get MyProxy parameters from OpenID and calls `logon()`.

Parameters `openid` – OpenID used to login at ESGF node.

```
malleefowl.esgf.logon.parse_openid(openid, ssl_verify=False)
```

parse openid document to get myproxy service

```
malleefowl.esgf.logon.cert_infos(filename)
```

malleefowl.esgf.search

Module Contents

```
malleefowl.esgf.search.LOGGER
```

```
malleefowl.esgf.search.date_from_filename(filename)
```

Example `cordex: tas_EUR-44i_ECMWF-ERAINT_evaluation_r1i1p1_HMS-ALADIN52_v1_mon_200101-200812.nc`

```
malleefowl.esgf.search.variable_filter(constraints, variables)
```

return True if variable fulfills constraints

```
malleefowl.esgf.search.temporal_filter(filename, start_date=None, end_date=None)
```

return True if file is in timerange start/end

```
class malleefowl.esgf.search.ESGSearch(url='http://localhost:8081/esg-search', distrib=False, replica=False, latest=True, monitor=None)
```

Bases: `object`

wrapper for esg search.

TODO: bbox constraint for datasets

```
show_status (self, message, progress)
```

```
search (self, constraints=[('project', 'CORDEX')], query=None, start=None, end=None, limit=1, offset=0, search_type='Dataset', temporal=False)
```

```
_index (self, datasets, limit, offset)
```

```
_file_context (self, dataset)
```

```
_aggregation_context (self, dataset)
```

```
threader (self)
```

```
_file_search_job (self, f_ctx, start_date, end_date)
```

```
_file_search (self, datasets, constraints, start_date, end_date)
```

```
_aggregation_search (self, datasets, constraints)
```


`malleefowl.processes`

Submodules

`malleefowl.processes.wps_download`

Module Contents

`malleefowl.processes.wps_download.LOGGER`

class `malleefowl.processes.wps_download.Download`
Bases: `pywps.Process`

The download process gets as input a list of URLs pointing to NetCDF files which should be downloaded.

The downloader first checks if the file is available in the local ESGF archive or cache. If not then the file will be downloaded and stored in a local cache. As a result it provides a list of local `file://` paths to the requested files.

The downloader does not download files if they are already in the ESGF archive or in the local cache.

`_handler` (*self, request, response*)

`malleefowl.processes.wps_esgsearch`

Module Contents

`malleefowl.processes.wps_esgsearch.LOGGER`

class `malleefowl.processes.wps_esgsearch.ESGSearchProcess`
Bases: `pywps.Process`

The ESGF search process runs a ESGF search request with constraints (project, experiment, ...) to get a list of matching files on ESGF data nodes. It is using `esgf-pyclient` Python client for the ESGF search API.

In addition to the `esgf-pyclient` the process checks if local replicas are available and would return the replica files instead of the original one.

The result is a JSON document with a list of `http://` URLs to files on ESGF data nodes.

TODO: bbox constraint for datasets

`_handler` (*self, request, response*)

`malleefowl.processes.wps_thredds`

Module Contents

class `malleefowl.processes.wps_thredds.ThreddsDownload`
Bases: `pywps.Process`

`_handler` (*self, request, response*)

`malleefowl.processes.wps_workflow`

Module Contents

`malleefowl.processes.wps_workflow.LOGGER`

class `malleefowl.processes.wps_workflow.DispelWorkflow`
Bases: `pywps.Process`

The workflow process is usually called by the [Phoenix](#) WPS web client to run WPS process for climate data (like cfchecker, climate indices with ocgis, ...) with a given selection of input data (currently NetCDF files from ESGF data nodes).

Currently the [Dispel4Py](#) workflow engine is used.

The Workflow for ESGF input data is as follows:

Search ESGF files -> Download ESGF files -> Run choosen process on local (downloaded) ESGF files.

`_handler` (*self, request, response*)

Package Contents

class `malleefowl.processes.ESGSearchProcess`
Bases: `pywps.Process`

The ESGF search process runs a ESGF search request with constraints (project, experiment, ...) to get a list of matching files on ESGF data nodes. It is using [esgf-pyclient](#) Python client for the ESGF search API.

In addition to the [esgf-pyclient](#) the process checks if local replicas are available and would return the replica files instead of the original one.

The result is a JSON document with a list of `http://` URLs to files on ESGF data nodes.

TODO: bbox constraint for datasets

`_handler` (*self, request, response*)

class `malleefowl.processes.Download`
Bases: `pywps.Process`

The download process gets as input a list of URLs pointing to NetCDF files which should be downloaded.

The downloader first checks if the file is available in the local ESGF archive or cache. If not then the file will be downloaded and stored in a local cache. As a result it provides a list of `file://` paths to the requested files.

The downloader does not download files if they are already in the ESGF archive or in the local cache.

`_handler` (*self, request, response*)

class `malleefowl.processes.ThreddsDownload`
Bases: `pywps.Process`

`_handler` (*self, request, response*)

class `malleefowl.processes.DispelWorkflow`
Bases: `pywps.Process`

The workflow process is usually called by the [Phoenix](#) WPS web client to run WPS process for climate data (like cfchecker, climate indices with ocgis, ...) with a given selection of input data (currently NetCDF files from ESGF data nodes).

Currently the [Dispel4Py](#) workflow engine is used.

The Workflow for ESGF input data is as follows:

Search ESGF files -> Download ESGF files -> Run choosen process on local (downloaded) ESGF files.

`__handler` (*self*, *request*, *response*)

`malleefowl.processes.processes`

`malleefowl.tests`

Submodules

`malleefowl.tests.common`

Module Contents

`malleefowl.tests.common.TESTDATA`

```
class malleefowl.tests.common.WpsTestClient
    Bases:pywps.tests.WpsClient
    get (self, *args, **kwargs)
```

`malleefowl.tests.common.client_for` (*service*)

`malleefowl.tests.test_download`

Module Contents

`malleefowl.tests.test_download.test_download()`

`malleefowl.tests.test_download.test_download_with_file_url()`

`malleefowl.tests.test_esgf_logon`

Module Contents

`malleefowl.tests.test_esgf_logon.test_parse_openid()`

`malleefowl.tests.test_esgf_search`

Module Contents

`malleefowl.tests.test_esgf_search.test_date_format()`

`malleefowl.tests.test_esgf_search.test_date_format_1900()`

```
class malleefowl.tests.test_esgf_search.EsgDistribSearchTestCase
    Bases:unittest.TestCase
    classmethod setUpClass (cls)
```

```
    test_file_cmip5_with_local_replica(self)
class malleefowl.tests.test_esgf_search.EsgSearchTestCase
    Bases:unittest.TestCase
    classmethod setUpClass(cls)
    test_dataset(self)
    test_file(self)
    test_file_cmip5_many(self)
    test_file_more_than_one(self)
    test_aggregation(self)
    test_file_cordex(self)
    test_file_cordex_date(self)
    test_file_cordex_many(self)
    test_file_cordex_fx(self)
    test_file_cordex_fly(self)
    test_file_cmip5(self)
    test_file_cmip5_date(self)
```

malleefowl.tests.test_utils

Module Contents

```
malleefowl.tests.test_utils.test_esgf_archive_path_cordex()
malleefowl.tests.test_utils.test_esgf_archive_path_cmip5()
malleefowl.tests.test_utils.test_esgf_archive_path_cmip5_noaa()
malleefowl.tests.test_utils.test_dupname()
malleefowl.tests.test_utils.test_user_id()
malleefowl.tests.test_utils.test_within_date_range()
malleefowl.tests.test_utils.test_filter_timesteps()
malleefowl.tests.test_utils.test_filter_timesteps2()
malleefowl.tests.test_utils.test_nc_copy()
```

malleefowl.tests.test_wps_caps

Module Contents

```
malleefowl.tests.test_wps_caps.test_wps_caps()
```

`malleefowl.tests.test_wps_download`

Module Contents

`malleefowl.tests.test_wps_download.test_wps_download()`

`malleefowl.tests.test_wps_esgsearch`

Module Contents

`malleefowl.tests.test_wps_esgsearch.test_dataset()`
`malleefowl.tests.test_wps_esgsearch.test_dataset_with_spaces()`
`malleefowl.tests.test_wps_esgsearch.test_dataset_out_of_limit()`
`malleefowl.tests.test_wps_esgsearch.test_dataset_out_of_offset()`
`malleefowl.tests.test_wps_esgsearch.test_dataset_latest()`
`malleefowl.tests.test_wps_esgsearch.test_dataset_query()`
`malleefowl.tests.test_wps_esgsearch.test_aggregation()`
`malleefowl.tests.test_wps_esgsearch.test_file()`

`malleefowl.tests.test_wps_thredds`

Module Contents

`malleefowl.tests.test_wps_thredds.test_wps_thredds_download()`

`malleefowl.tests.test_wps_workflow`

Module Contents

`malleefowl.tests.test_wps_workflow.test_wps_thredds_workflow()`

6.1.2 Submodules

`malleefowl.config`

Module Contents

`malleefowl.config.LOGGER`
`malleefowl.config.DEFAULT_NODE = default`
`malleefowl.config.DKRZ_NODE = dkrz`
`malleefowl.config.IPSL_NODE = ipsl`
`malleefowl.config.wps_url()`

```
malleefowl.config.cache_path()
malleefowl.config.archive_root()
malleefowl.config.archive_node()
```

malleefowl.demo

Module Contents

```
malleefowl.demo.LOGGER
malleefowl.demo.get_host()
malleefowl.demo._run(application, daemon=False)
malleefowl.demo.main()
```

malleefowl.download

TODO: handle parallel downloads

Module Contents

```
malleefowl.download.LOGGER
malleefowl.download.download_with_archive(url, credentials=None)
    Downloads file. Checks before downloading if file is already in local esgf archive.
malleefowl.download.download(url, use_file_url=False, credentials=None)
    Downloads url and returns local filename.
```

Parameters

- **url** – url of file
- **use_file_url** – True if result should be a file url “file://”, otherwise use system path.
- **credentials** – path to credentials if security is needed to download file

Returns downloaded file with either `file://` or system path

```
malleefowl.download.wget(url, use_file_url=False, credentials=None)
    Downloads url and returns local filename.
```

TODO: refactor cache handling.

Parameters

- **url** – url of file
- **use_file_url** – True if result should be a file url “file://”, otherwise use system path.
- **credentials** – path to credentials if security is needed to download file

Returns downloaded file with either `file://` or system path

```
malleefowl.download.download_files(urls=[], credentials=None, monitor=None)
malleefowl.download.download_files_from_thredds(url, recursive=False, monitor=None)
```

```

class malleefowl.download.DownloadManager (monitor=None)
    Bases:object

    show_status (self, message, progress)

    threader (self)

    download_job (self, url, credentials)

    download (self, urls, credentials=None)

```

malleefowl.exceptions

Module Contents

```

exception malleefowl.exceptions.ProcessFailed
    Bases:exceptions.Exception

```

malleefowl.utils

Utility functions for WPS processes.

Module Contents

```

malleefowl.utils.LOGGER

malleefowl.utils.esgf_archive_path (url)

malleefowl.utils.dupname (path, filename)
    avoid dupliate filenames TODO: needs to be improved

malleefowl.utils.user_id (openid)
    generate user_id from openid

malleefowl.utils.within_date_range (timesteps, start=None, end=None)

malleefowl.utils.filter_timesteps (timesteps, aggregation='monthly', start=None, end=None)

malleefowl.utils.nc_copy (source, target, overwrite=True, time_dimname='time', nchunk=10, is-
                           tart=0, istop=-1, format='NETCDF3_64BIT')
    copy netcdf file from.opendap to netcdf3 file

```

Parameters

- **overwrite** – Overwrite destination file (default is to raise an error if output file already exists).
- **format** – netcdf3 format to use (NETCDF3_64BIT by default, can be set to NETCDF3_CLASSIC)
- **chunk** – number of records along unlimited dimension to write at once. Default 10. Ignored if there is no unlimited dimension. chunk=0 means write all the data at once.
- **istart** – number of record to start at along unlimited dimension. Default 0. Ignored if there is no unlimited dimension.
- **istop** – number of record to stop at along unlimited dimension. Default -1. Ignored if there is no unlimited dimension.

malleefowl.workflow

Module Contents

malleefowl.workflow.**LOGGER**

class malleefowl.workflow.**MonitorPE** (*output=None*)

Bases:dispel4py.base.BasePE

INPUT_NAME = *input*

OUTPUT_NAME = *output*

set_monitor (*self, monitor, start_progress=0, end_progress=100*)

class malleefowl.workflow.**GenericWPS** (*url, identifier, resource='resource', inputs=[], output=None, headers=None*)

Bases:malleefowl.workflow.MonitorPE

STATUS_NAME = *status*

STATUS_LOCATION_NAME = *status_location*

progress (*self, execution*)

monitor_execution (*self, execution*)

_build_wps_inputs (*self*)

_build_wps_outputs (*self*)

execute (*self*)

_set_inputs (*self, inputs*)

process (*self, inputs*)

_process (*self, inputs*)

class malleefowl.workflow.**EsgSearch** (*url, search_url='https://esgf-data.dkrz.de/esg-search', constraints='project:CORDEX', query=None, limit=100, search_type='File', distrib=False, replica=False, latest=True, temporal=False, start=None, end=None*)

Bases:malleefowl.workflow.GenericWPS

_process (*self, inputs*)

class malleefowl.workflow.**SolrSearch** (*url, query, filter_query=None*)

Bases:malleefowl.workflow.MonitorPE

Run search against birdhouse solr index and return a list of download urls.

process (*self, inputs*)

class malleefowl.workflow.**Download** (*url, headers=None*)

Bases:malleefowl.workflow.GenericWPS

_process (*self, inputs*)

class malleefowl.workflow.**ThreddsDownload** (*url, catalog_url, headers=None*)

Bases:malleefowl.workflow.GenericWPS

_process (*self, inputs*)

malleefowl.workflow.**esgf_workflow** (*source, worker, monitor=None, headers=None*)


```
malleefowl.workflow.thredds_workflow(source, worker, monitor=None, headers=None)
malleefowl.workflow.solr_workflow(source, worker, monitor=None, headers=None)
malleefowl.workflow.run(workflow, monitor=None, headers=None)
```

malleefowl.wsgi

Module Contents

```
malleefowl.wsgi.create_app(cfgfiles=None)
malleefowl.wsgi.application
```

6.1.3 Package Contents

```
malleefowl.application
malleefowl.main()
```


CHAPTER 7

Indices and tables

- `genindex`
- `modindex`
- `search`

m

- malleefowl, [19](#)
- malleefowl.config, [25](#)
- malleefowl.demo, [26](#)
- malleefowl.download, [26](#)
- malleefowl.esgf, [19](#)
- malleefowl.esgf.logon, [19](#)
- malleefowl.esgf.search, [20](#)
- malleefowl.exceptions, [27](#)
- malleefowl.processes, [21](#)
- malleefowl.processes.wps_download, [21](#)
- malleefowl.processes.wps_esgsearch, [21](#)
- malleefowl.processes.wps_thredds, [21](#)
- malleefowl.processes.wps_workflow, [22](#)
- malleefowl.tests, [23](#)
- malleefowl.tests.common, [23](#)
- malleefowl.tests.test_download, [23](#)
- malleefowl.tests.test_esgf_logon, [23](#)
- malleefowl.tests.test_esgf_search, [23](#)
- malleefowl.tests.test_utils, [24](#)
- malleefowl.tests.test_wps_caps, [24](#)
- malleefowl.tests.test_wps_download, [25](#)
- malleefowl.tests.test_wps_esgsearch, [25](#)
- malleefowl.tests.test_wps_thredds, [25](#)
- malleefowl.tests.test_wps_workflow, [25](#)
- malleefowl.utils, [27](#)
- malleefowl.workflow, [28](#)
- malleefowl.wsgi, [29](#)

Symbols

_aggregation_context() (malleefowl.esgf.search.ESGSearch method), 20
 _aggregation_search() (malleefowl.esgf.search.ESGSearch method), 20
 _build_wps_inputs() (malleefowl.workflow.GenericWPS method), 28
 _build_wps_outputs() (malleefowl.workflow.GenericWPS method), 28
 _file_context() (malleefowl.esgf.search.ESGSearch method), 20
 _file_search() (malleefowl.esgf.search.ESGSearch method), 20
 _file_search_job() (malleefowl.esgf.search.ESGSearch method), 20
 _handler() (malleefowl.processes.DispelWorkflow method), 23
 _handler() (malleefowl.processes.Download method), 22
 _handler() (malleefowl.processes.ESGSearchProcess method), 22
 _handler() (malleefowl.processes.ThreddsDownload method), 22
 _handler() (malleefowl.processes.wps_download.Download method), 21
 _handler() (malleefowl.processes.wps_esgsearch.ESGSearchProcess method), 21
 _handler() (malleefowl.processes.wps_thredds.ThreddsDownload method), 21
 _handler() (malleefowl.processes.wps_workflow.DispelWorkflow method), 22
 _index() (malleefowl.esgf.search.ESGSearch method), 20
 _process() (malleefowl.workflow.Download method), 28
 _process() (malleefowl.workflow.EsgSearch method), 28
 _process() (malleefowl.workflow.GenericWPS method), 28
 _process() (malleefowl.workflow.ThreddsDownload method), 28
 _run() (in module malleefowl.demo), 26
 _set_inputs() (malleefowl.workflow.GenericWPS

method), 28

A

application (in module malleefowl), 29
 application (in module malleefowl.wsgi), 29
 archive_node() (in module malleefowl.config), 26
 archive_root() (in module malleefowl.config), 26

C

cache_path() (in module malleefowl.config), 25
 cert_infos() (in module malleefowl.esgf.logon), 20
 client_for() (in module malleefowl.tests.common), 23
 create_app() (in module malleefowl.wsgi), 29

D

date_from_filename() (in module malleefowl.esgf.search), 20
 DEFAULT_NODE (in module malleefowl.config), 25
 DispelWorkflow (class in malleefowl.processes), 22
 DispelWorkflow (class in malleefowl.processes.wps_workflow), 22
 DKRZ_NODE (in module malleefowl.config), 25
 Download (class in malleefowl.processes), 22
 Download (class in malleefowl.processes.wps_download), 21
 Download (class in malleefowl.workflow), 28
 download() (in module malleefowl.download), 26
 download() (malleefowl.download.DownloadManager method), 27
 download_files() (in module malleefowl.download), 26
 download_files_from_thredds() (in module malleefowl.download), 26
 download_job() (malleefowl.download.DownloadManager method), 27
 download_with_archive() (in module malleefowl.download), 26
 DownloadManager (class in malleefowl.download), 26
 dupname() (in module malleefowl.utils), 27

E

EsgDistribSearchTestCase (class in malleefowl.tests.test_esgf_search), 23
 esgf_archive_path() (in module malleefowl.utils), 27
 esgf_workflow() (in module malleefowl.workflow), 28
 ESGSearch (class in malleefowl.esgf.search), 20
 EsgSearch (class in malleefowl.workflow), 28
 ESGSearchProcess (class in malleefowl.processes), 22
 ESGSearchProcess (class in malleefowl.processes.wps_esgsearch), 21
 EsgSearchTestCase (class in malleefowl.tests.test_esgf_search), 24
 execute() (malleefowl.workflow.GenericWPS method), 28

F

filter_timesteps() (in module malleefowl.utils), 27

G

GenericWPS (class in malleefowl.workflow), 28
 get() (malleefowl.tests.common.WpsTestClient method), 23
 get_host() (in module malleefowl.demo), 26

I

INPUT_NAME (malleefowl.workflow.MonitorPE attribute), 28
 IPSL_NODE (in module malleefowl.config), 25

L

LOGGER (in module malleefowl.config), 25
 LOGGER (in module malleefowl.demo), 26
 LOGGER (in module malleefowl.download), 26
 logger (in module malleefowl.esgf.logon), 19
 LOGGER (in module malleefowl.esgf.search), 20
 LOGGER (in module malleefowl.processes.wps_download), 21
 LOGGER (in module malleefowl.processes.wps_esgsearch), 21
 LOGGER (in module malleefowl.processes.wps_workflow), 22
 LOGGER (in module malleefowl.utils), 27
 LOGGER (in module malleefowl.workflow), 28

M

main() (in module malleefowl), 29
 main() (in module malleefowl.demo), 26
 malleefowl (module), 19
 malleefowl.config (module), 25
 malleefowl.demo (module), 26
 malleefowl.download (module), 26
 malleefowl.esgf (module), 19
 malleefowl.esgf.logon (module), 19

malleefowl.esgf.search (module), 20
 malleefowl.exceptions (module), 27
 malleefowl.processes (module), 21
 malleefowl.processes.wps_download (module), 21
 malleefowl.processes.wps_esgsearch (module), 21
 malleefowl.processes.wps_thredds (module), 21
 malleefowl.processes.wps_workflow (module), 22
 malleefowl.tests (module), 23
 malleefowl.tests.common (module), 23
 malleefowl.tests.test_download (module), 23
 malleefowl.tests.test_esgf_logon (module), 23
 malleefowl.tests.test_esgf_search (module), 23
 malleefowl.tests.test_utils (module), 24
 malleefowl.tests.test_wps_caps (module), 24
 malleefowl.tests.test_wps_download (module), 25
 malleefowl.tests.test_wps_esgsearch (module), 25
 malleefowl.tests.test_wps_thredds (module), 25
 malleefowl.tests.test_wps_workflow (module), 25
 malleefowl.utils (module), 27
 malleefowl.workflow (module), 28
 malleefowl.wsgi (module), 29
 monitor_execution() (malleefowl.workflow.GenericWPS method), 28
 MonitorPE (class in malleefowl.workflow), 28
 myproxy_logon_with_openid() (in module malleefowl.esgf.logon), 19

N

nc_copy() (in module malleefowl.utils), 27

O

OUTPUT_NAME (malleefowl.workflow.MonitorPE attribute), 28

P

parse_openid() (in module malleefowl.esgf.logon), 20
 process() (malleefowl.workflow.GenericWPS method), 28
 process() (malleefowl.workflow.SolrSearch method), 28
 processes (in module malleefowl.processes), 23
 ProcessFailed, 27
 progress() (malleefowl.workflow.GenericWPS method), 28

R

run() (in module malleefowl.workflow), 29

S

search() (malleefowl.esgf.search.ESGSearch method), 20
 set_monitor() (malleefowl.workflow.MonitorPE method), 28
 setUpClass() (malleefowl.tests.test_esgf_search.EsgDistribSearchTestCase class method), 23

[setUpClass\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase class method\), 24](#)
[show_status\(\) \(malleefowl.download.DownloadManager method\), 27](#)
[show_status\(\) \(malleefowl.esgf.search.ESGSearch method\), 20](#)
[solr_workflow\(\) \(in module malleefowl.workflow\), 29](#)
[SolrSearch \(class in malleefowl.workflow\), 28](#)
[STATUS_LOCATION_NAME \(malleefowl.workflow.GenericWPS attribute\), 28](#)
[STATUS_NAME \(malleefowl.workflow.GenericWPS attribute\), 28](#)

T

[temporal_filter\(\) \(in module malleefowl.esgf.search\), 20](#)
[test_aggregation\(\) \(in module malleefowl.tests.test_wps_esgsearch\), 25](#)
[test_aggregation\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase method\), 24](#)
[test_dataset\(\) \(in module malleefowl.tests.test_wps_esgsearch\), 25](#)
[test_dataset\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase method\), 24](#)
[test_dataset_latest\(\) \(in module malleefowl.tests.test_wps_esgsearch\), 25](#)
[test_dataset_out_of_limit\(\) \(in module malleefowl.tests.test_wps_esgsearch\), 25](#)
[test_dataset_out_of_offset\(\) \(in module malleefowl.tests.test_wps_esgsearch\), 25](#)
[test_dataset_query\(\) \(in module malleefowl.tests.test_wps_esgsearch\), 25](#)
[test_dataset_with_spaces\(\) \(in module malleefowl.tests.test_wps_esgsearch\), 25](#)
[test_date_format\(\) \(in module malleefowl.tests.test_esgf_search\), 23](#)
[test_date_format_1900\(\) \(in module malleefowl.tests.test_esgf_search\), 23](#)
[test_download\(\) \(in module malleefowl.tests.test_download\), 23](#)
[test_download_with_file_url\(\) \(in module malleefowl.tests.test_download\), 23](#)
[test_dupname\(\) \(in module malleefowl.tests.test_utils\), 24](#)
[test_esgf_archive_path_cmip5\(\) \(in module malleefowl.tests.test_utils\), 24](#)
[test_esgf_archive_path_cmip5_noaa\(\) \(in module malleefowl.tests.test_utils\), 24](#)
[test_esgf_archive_path_cordex\(\) \(in module malleefowl.tests.test_utils\), 24](#)
[test_file\(\) \(in module malleefowl.tests.test_wps_esgsearch\), 25](#)
[test_file\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase method\), 24](#)
[test_file_cmip5\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase method\), 24](#)
[test_file_cmip5_date\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase method\), 24](#)
[test_file_cmip5_many\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase method\), 24](#)
[test_file_cmip5_with_local_replica\(\) \(malleefowl.tests.test_esgf_search.EsgDistribSearchTestCase method\), 23](#)
[test_file_cordex\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase method\), 24](#)
[test_file_cordex_date\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase method\), 24](#)
[test_file_cordex_fly\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase method\), 24](#)
[test_file_cordex_fx\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase method\), 24](#)
[test_file_cordex_many\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase method\), 24](#)
[test_file_more_than_one\(\) \(malleefowl.tests.test_esgf_search.EsgSearchTestCase method\), 24](#)
[test_filter_timesteps\(\) \(in module malleefowl.tests.test_utils\), 24](#)
[test_filter_timesteps2\(\) \(in module malleefowl.tests.test_utils\), 24](#)
[test_nc_copy\(\) \(in module malleefowl.tests.test_utils\), 24](#)
[test_parse_openid\(\) \(in module malleefowl.tests.test_esgf_logon\), 23](#)
[test_user_id\(\) \(in module malleefowl.tests.test_utils\), 24](#)
[test_within_date_range\(\) \(in module malleefowl.tests.test_utils\), 24](#)
[test_wps_caps\(\) \(in module malleefowl.tests.test_wps_caps\), 24](#)
[test_wps_download\(\) \(in module malleefowl.tests.test_wps_download\), 25](#)
[test_wps_thredds_download\(\) \(in module malleefowl.tests.test_wps_thredds\), 25](#)
[test_wps_thredds_workflow\(\) \(in module malleefowl.tests.test_wps_workflow\), 25](#)
[TESTDATA \(in module malleefowl.tests.common\), 23](#)
[threader\(\) \(malleefowl.download.DownloadManager method\), 27](#)
[threader\(\) \(malleefowl.esgf.search.ESGSearch method\), 20](#)
[thredds_workflow\(\) \(in module malleefowl.workflow\), 28](#)
[ThreddsDownload \(class in malleefowl.processes\), 22](#)
[ThreddsDownload \(class in malleefowl.processes.wps_thredds\), 21](#)
[ThreddsDownload \(class in malleefowl.workflow\), 28](#)

U

[use_id\(\) \(in module malleefowl.utils\), 27](#)

V

`variable_filter()` (in module `malleefowl.esgf.search`), [20](#)

W

`wget()` (in module `malleefowl.download`), [26](#)

`within_date_range()` (in module `malleefowl.utils`), [27](#)

`wps_url()` (in module `malleefowl.config`), [25](#)

`WpsTestClient` (class in `malleefowl.tests.common`), [23](#)