DEEPaaS Documentation

Release 1.0.2

DEEP-Hybrid-DataCloud consortium

Nov 30, 2021

Contents

Release v1.0.2. (*Installation*)

Orpy is a client library for the INDIGO-DataCloud orchestrator REST API. There's a *Python API* (the `orpy.client` module), and a *command-line script* (installed as **`orpy`**). Each implements the entire INDIGO-DataCloud orchestrator REST API.

Contents:

CHAPTER 1

Installation

At the command line:

```
$ pip install orpy
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv orpy
$ pip install orpy
```

Usage:

CHAPTER 2

orpy CLI application

2.1 Authentication

In order to interact with the INDIGO PaaS Orchestrator we need to use an OpenID Connect access token from a trusted OpenID Connect provider at the orchestrator.

Please either store your access token in ORCHESTRATOR_TOKEN or set the account to use with **oidc-agent** in the OIDC_ACCOUNT and the socket path of the oidc-agent in the OIDC_SOCK environment variable:

```
export ORCHESTRATOR_TOKEN=<your access token>
   OR
export OIDC_SOCK=<path to the oidc-agent socket>
export OIDC_ACCOUNT=<account to use>
```

Usually, the OIDC_SOCK environmental variable is already exported if you are using **oidc-agent**.

As an alternative, you can pass the socket path and the account through the command line with the --oidc-agent-sock and --oidc-agent-account parameters.

2.2 Usage

Command line client for the INDIGO PaaS Orchestrator.

> Please, before using this command put your a valid OpenID Connnect access token into the ORCHESTRATOR_TOKEN environment variable, so that we can use this token for authentication.

```
orpy
    [--version]
    [-v | -q]
    [--log-file LOG_FILE]
    [--debug]
    [--oidc-agent-sock <oidc-agent-socket>]
```

(continues on next page)

(continued from previous page)

```
    [--oidc-agent-account <oidc-agent-account>]
    [--url <orchestrator-url>]
```

--version
show program's version number and exit

-v, --verbose
Increase verbosity of output. Can be repeated.

-q, --quiet
Suppress output except warnings and errors.

--log-file <LOG_FILE>
Specify a file to log output. Disabled by default.

--debug
Show tracebacks on errors.

--oidc-agent-sock <oidc-agent-socket>
The path for the oidc-agent socket to use to get and renew access tokens from the OpenID Connect provider. This defaults to the OIDC_SOCK environment variable, that should be automatically set up if you are using oidc-agent. In order to use the oidc-agent you must also pass the –oidc-agent-account parameter, or set the OIDC_ACCOUNT environment variable.

--oidc-agent-account <oidc-agent-account>
The oidc-agent account that we will use to get tokens from. In order to use the oidc-agent you must pass thos parameter or set the OIDC_ACCOUNT environment variable.

--url <orchestrator-url>
The base url of the orchestrator rest interface. Alternative the environment variable ORCHESTRATOR_URL can be used.

Authentication:

In order to interact with the INDIGO PaaS Orchestrator we need to use an OpenID Connect access token from a trusted OpenID Connect provider at the orchestrator.

Please either store your access token in 'ORCHESTRATOR_TOKEN' or set the account to use with oidc-agent in the 'OIDC_ACCOUNT' and the socket path of the oidc-agent in the 'OIDC_SOCK' environment variable:

export ORCHESTRATOR_TOKEN=<your access token> OR

export OIDC_SOCK=<path to the oidc-agent socket> export OIDC_ACCOUNT=<account to use>

Usually, the OIDC_SOCK environmental variable is already exported if you are using oidc-agent.

As an alternative, you can pass the socket path and the account through the command line with the –oidc-agent-sock and –oidc-agent-account parameters.

2.3 List of commands

2.3.1 Query orchestrator information

2.3.2 Manage deployments

deployment create

Create a deployment.

```
orpy deployment create
    [-f {json,shell,table,value,yaml}]
    [-c COLUMN]
    [--noindent]
    [--prefix PREFIX]
    [--max-width <integer>]
    [--fit-width]
    [--print-empty]
    [--callback-url CALLBACK]
    [--max-providers-retry MAX_RETRIES]
    [--keep-last-attempt KEEP_LAST]
    <template
    file>
    [<parameter>=<value> [<parameter>=<value> ...]]
```

-f <FORMATTER>, **--format** <FORMATTER>
the output format, defaults to table

-c COLUMN, **--column** COLUMN
specify the column(s) to include, can be repeated to show multiple columns

--noindent
whether to disable indenting the JSON

--prefix <PREFIX>
add a prefix to all variable names

--max-width <integer>
Maximum display width, <1 to disable. You can also use the CLIFF_MAX_TERM_WIDTH environment variable, but the parameter takes precedence.

--fit-width
Fit the table to the display width. Implied if –max-width greater than 0. Set the environment variable CLIFF_FIT_WIDTH=1 to always enable

--print-empty
Print empty table if there is no data to show.

--callback-url <CALLBACK>
The callback url.

--max-providers-retry <MAX_RETRIES>
Maximum number of cloud providers to be used in case of failure (Default is to be unbounded).

--keep-last-attempt <KEEP_LAST>
In case of failure, keep the resources allocated in the last try (Default: True, accepts boolean values).

template file
TOSCA template file.

parameter>=<value
Input parameter for the deployment in the form <parameter>=<value>. Can be specified several times.

This command is provided by the orpy plugin.

deployment delete

Show details about an existing deployment.

```
orpy deployment delete <deployment uuid>
```

deployment uuid
Deployment UUID to delete.

This command is provided by the orpy plugin.

deployment list

List existing deployments at orchestrator.

```
orpy deployment list
    [-f {csv,json,table,value,yaml}]
    [-c COLUMN]
    [--quote {all,minimal,none,nonnumeric}]
    [--noindent]
    [--max-width <integer>]
    [--fit-width]
    [--print-empty]
    [--sort-column SORT_COLUMN]
    [--sort-ascending | --sort-descending]
```

-f <FORMATTER>, **--format** <FORMATTER>
the output format, defaults to table

-c COLUMN, **--column** COLUMN
specify the column(s) to include, can be repeated to show multiple columns

--quote <QUOTE_MODE>
when to include quotes, defaults to nonnumeric

--noindent
whether to disable indenting the JSON

--max-width <integer>
Maximum display width, <1 to disable. You can also use the CLIFF_MAX_TERM_WIDTH environment variable, but the parameter takes precedence.

--fit-width
Fit the table to the display width. Implied if –max-width greater than 0. Set the environment variable CLIFF_FIT_WIDTH=1 to always enable

--print-empty
Print empty table if there is no data to show.

--sort-column SORT_COLUMN
specify the column(s) to sort the data (columns specified first have a priority, non-existing columns are ignored), can be repeated

--sort-ascending
sort the column(s) in ascending order

--sort-descending
sort the column(s) in descending order

This command is provided by the orpy plugin.

deployment show

Show details about an existing deployment.

```
orpy deployment show
    [-f {json,shell,table,value,yaml}]
    [-c COLUMN]
    [--noindent]
    [--prefix PREFIX]
    [--max-width <integer>]
    [--fit-width]
    [--print-empty]
    [-l]
    <deployment
    uuid>
```

-f <FORMATTER>, **--format** <FORMATTER>
: the output format, defaults to table

-c COLUMN, **--column** COLUMN
: specify the column(s) to include, can be repeated to show multiple columns

--noindent
: whether to disable indenting the JSON

--prefix <PREFIX>
: add a prefix to all variable names

--max-width <integer>
: Maximum display width, <1 to disable. You can also use the CLIFF_MAX_TERM_WIDTH environment variable, but the parameter takes precedence.

--fit-width
: Fit the table to the display width. Implied if –max-width greater than 0. Set the environment variable CLIFF_FIT_WIDTH=1 to always enable

--print-empty
: Print empty table if there is no data to show.

-l, --long
: Show additional fields in output.

deployment uuid
: Deployment UUID to show.

This command is provided by the orpy plugin.

deployment template

Get template used for a given deployment.

```
orpy deployment template
    [-f {json,shell,table,value,yaml}]
    [-c COLUMN]
    [--noindent]
    [--prefix PREFIX]
    [--max-width <integer>]
    [--fit-width]
    [--print-empty]
```

(continues on next page)

(continued from previous page)

```
    <deployment
    uuid>
```

-f <FORMATTER>, **--format** <FORMATTER>
the output format, defaults to table

-c COLUMN, **--column** COLUMN
specify the column(s) to include, can be repeated to show multiple columns

--noindent
whether to disable indenting the JSON

--prefix <PREFIX>
add a prefix to all variable names

--max-width <integer>
Maximum display width, <1 to disable. You can also use the CLIFF_MAX_TERM_WIDTH environment variable, but the parameter takes precedence.

--fit-width
Fit the table to the display width. Implied if –max-width greater than 0. Set the environment variable CLIFF_FIT_WIDTH=1 to always enable

--print-empty
Print empty table if there is no data to show.

deployment uuid
Deployment UUID to get template for.

This command is provided by the orpy plugin.

deployment update

Update an existing deployment.

```
orpy deployment update
    [-f {json,shell,table,value,yaml}]
    [-c COLUMN]
    [--noindent]
    [--prefix PREFIX]
    [--max-width <integer>]
    [--fit-width]
    [--print-empty]
    [--callback-url CALLBACK]
    [--max-providers-retry MAX_RETRIES]
    [--keep-last-attempt KEEP_LAST]
    <deployment
    uuid>
    <template
    file>
    [<parameter>=<value> [<parameter>=<value> ...]]
```

-f <FORMATTER>, **--format** <FORMATTER>
the output format, defaults to table

-c COLUMN, **--column** COLUMN
specify the column(s) to include, can be repeated to show multiple columns

--noindent
whether to disable indenting the JSON

--prefix <PREFIX>
add a prefix to all variable names

--max-width <integer>
Maximum display width, <1 to disable. You can also use the CLIFF_MAX_TERM_WIDTH environment variable, but the parameter takes precedence.

--fit-width
Fit the table to the display width. Implied if –max-width greater than 0. Set the environment variable CLIFF_FIT_WIDTH=1 to always enable

--print-empty
Print empty table if there is no data to show.

--callback-url <CALLBACK>
The callback url.

--max-providers-retry <MAX_RETRIES>
Maximum number of cloud providers to be used in case of failure (Default is to be unbounded).

--keep-last-attempt <KEEP_LAST>
In case of failure, keep the resources allocated in the last try (Default: True, accepts boolean values).

deployment uuid
Deployment UUID to update.

template file
TOSCA template file.

parameter>=<value
Input parameter for the deployment in the form <parameter>=<value>. Can be specified several times.

This command is provided by the orpy plugin.

2.3.3 Manage resources

resource list

List Resources for a given deployment.

```
orpy resource list
    [-f {csv,json,table,value,yaml}]
    [-c COLUMN]
    [--quote {all,minimal,none,nonnumeric}]
    [--noindent]
    [--max-width <integer>]
    [--fit-width]
    [--print-empty]
    [--sort-column SORT_COLUMN]
    [--sort-ascending | --sort-descending]
    <deployment
    uuid>
```

-f <FORMATTER>, **--format** <FORMATTER>
the output format, defaults to table

-c COLUMN, **--column** COLUMN
specify the column(s) to include, can be repeated to show multiple columns

--quote <QUOTE_MODE>
when to include quotes, defaults to nonnumeric

--noindent
whether to disable indenting the JSON

--max-width <integer>
Maximum display width, <1 to disable. You can also use the CLIFF_MAX_TERM_WIDTH environment variable, but the parameter takes precedence.

--fit-width
Fit the table to the display width. Implied if –max-width greater than 0. Set the environment variable CLIFF_FIT_WIDTH=1 to always enable

--print-empty
Print empty table if there is no data to show.

--sort-column SORT_COLUMN
specify the column(s) to sort the data (columns specified first have a priority, non-existing columns are ignored), can be repeated

--sort-ascending
sort the column(s) in ascending order

--sort-descending
sort the column(s) in descending order

deployment uuid
Deployment UUID to show.

This command is provided by the orpy plugin.

resource show

Show details about a resource for a given deployment.

```
orpy resource show
    [-f {json,shell,table,value,yaml}]
    [-c COLUMN]
    [--noindent]
    [--prefix PREFIX]
    [--max-width <integer>]
    [--fit-width]
    [--print-empty]
    <deployment
    uuid>
    <resource
    uuid>
```

-f <FORMATTER>, **--format** <FORMATTER>
the output format, defaults to table

-c COLUMN, **--column** COLUMN
specify the column(s) to include, can be repeated to show multiple columns

--noindent
whether to disable indenting the JSON

--prefix <PREFIX>
add a prefix to all variable names

--max-width <integer>
Maximum display width, <1 to disable. You can also use the CLIFF_MAX_TERM_WIDTH environment variable, but the parameter takes precedence.

--fit-width
Fit the table to the display width. Implied if –max-width greater than 0. Set the environment variable CLIFF_FIT_WIDTH=1 to always enable

--print-empty
Print empty table if there is no data to show.

deployment uuid
Deployment UUID for the resource.

resource uuid
Resource UUID to show.

This command is provided by the orpy plugin.

CHAPTER 3

Orpy API bindings

You can use this library to interact with the INDIGO PaaS Orchestrator:

```
>>> from orpy.client import client
>>> orpy = client.OrpyClient(
...     url=ORCHESTRATOR_URL,
...     token=ORCHESTRATOR_TOKEN)
>>> deployments = orpy.deployments.list()
>>> deployments[0]
<Deployment cloudProviderName=provider-BARI, createdBy={u'subject': u'de28e179-ec86-
↪4915-a748-7a37f8d80311', u'issuer': u'https://iam.deep-hybrid-datacloud.eu/'},␣
↪creationTime=2019-05-27T11:31+0000, links=[{u'href': u'https://paas.cloud.cnaf.infn.
↪it/orchestrator/deployments/11e98073-06f3-6797-9258-0242ac140005', u'rel': u'self'},
↪ {u'href': u'https://paas.cloud.cnaf.infn.it/orchestrator/deployments/11e98073-06f3-
↪6797-9258-0242ac140005/resources', u'rel': u'resources'}, {u'href': u'https://paas.
↪cloud.cnaf.infn.it/orchestrator/deployments/11e98073-06f3-6797-9258-0242ac140005/
↪template', u'rel': u'template'}], outputs={}, physicalId=11e98073-06f3-6797-9258-
↪0242ac140005, status=CREATE_FAILED, statusReason=Error while checking the␣
↪deployment status; nested exception is feign.RetryableException: mesos.ui.sav.sk␣
↪executing GET https://mesos.ui.sav.sk/marathon/v2/groups/11e98073-06f3-6797-9258-
↪0242ac140005, task=NONE, updateTime=2019-05-29T02:05+0000, uuid=11e98073-06f3-6797-
↪9258-0242ac140005>
>>> deployments[0].status
CREATE_FAILED
```

3.1 Orpy Client

Use this to interact with the INDIGO-DatatCloud orchestrator.

class orpy.client.client.**OrpyClient**(*url*, *oidc_agent=None*, *token=None*, *debug=False*)

An INDIGO-DataCloud PaaS orchestrator client class.

config

Interface to query for Orchestrator configuration.

Returns Configuration interface.

Return type orpy.info.Config

delete(*url*, ***kwargs*)
Perform a DELETE request.

This calls *request()* with `method` set to `DELETE`.

deployments
Interface to query for deployments.

Returns Deployments interface.

Return type *orpy.client.deployments.Deployments*

get(*url*, ***kwargs*)
Perform a GET request.

This calls *request()* with `method` set to `GET`.

head(*url*, ***kwargs*)
Perform a HEAD request.

This calls *request()* with `method` set to `HEAD`.

info
Interface to query for Orchestrator information.

Returns Information interface.

Return type orpy.info.Info

patch(*url*, ***kwargs*)
Perform a PATCH request.

This calls *request()* with `method` set to `PATCH`.

post(*url*, ***kwargs*)
Perform a POST request.

This calls *request()* with `method` set to `POST`.

put(*url*, ***kwargs*)
Perform a PUT request.

This calls *request()* with `method` set to `PUT`.

request(*url*, *method*, *json=None*, ***kwargs*)
Send an HTTP request with the specified characteristics.

Wrapper around *requests.Session.request* to handle tasks such as setting headers, JSON encoding/decoding, and error handling.

Arguments that are not handled are passed through to the requests library.

Parameters

- **url** (*str*) – Path or fully qualified URL of the HTTP request. If only a path is provided then the URL will be prefixed with the attribute self.url. If a fully qualified URL is provided then self.url will be ignored.
- **method** (*str*) – The http method to use. (e.g. 'GET', 'POST')
- **json** – Some data to be represented as JSON. (optional)

- **kwargs** – any other parameter that can be passed to `requests.Session.request()` (such as *headers*). Except:
  - *data* will be overwritten by the data in the *json* param.
  - *allow_redirects* is ignored as redirects are handled by the session.

Returns The response to the request.

resources
Interface to query for resources.

Returns Resources interface.

Return type *orpy.client.resources.Resources*

3.2 Deployments interface

class orpy.client.deployments.**Deployments**(*client*)
Manage Orchestrator deployments.

create(*template*, *callback_url=None*, *max_providers_retry=None*, *keep_last_attemp=True*, *parameters={}*)
Create a deployment.

Parameters

- **template** (*str*) – The TOSCA template to use.
- **callback_url** (*str*) – The orchestrator callback url.
- **max_providers_retry** (*int*) – Maximum number of providers to retry.
- **keep_last_attemp** (*bool*) – Whether to keep the allocated resources in case of failure.

Returns The created deployment

Return type *orpy.client.base.Deployment*

delete(*uuid*)
Delete a deployment.

Parameters **uuid** (*str*) – The UUID of the deployment to delete.

Returns None

Return type None

get_template(*uuid*)
Get the TOSCA template of a deployment.

Parameters **uuid** (*str*) – The UUID of the deployment.

Returns The TOSCA template for the deployment

Return type *orpy.client.base.TOSCATemplate*

list()
List existing deployments.

Returns List of orpy.client.base.Deployment

Return type list

show(*uuid*)
Show details about a deployment.

Parameters **uuid** (*str*) – The UUID of the deployment to show.

Returns The deployment requested

Return type *orpy.client.base.Deployment*

update(*uuid*, *template*, *callback_url=None*, *max_providers_retry=None*, *keep_last_attemp=True*, *parameters={}*)
Update a deployment.

Parameters

- **uuid** (*str*) – The UUID of the deployment.
- **template** (*str*) – The TOSCA template to use.
- **callback_url** (*str*) – The orchestrator callback url.
- **max_providers_retry** (*int*) – Maximum number of providers to retry.
- **keep_last_attemp** (*bool*) – Whether to keep the allocated resources in case of failure.

Returns The updated deployment

Return type *orpy.client.base.Deployment*

3.3 Resources interface

class orpy.client.resources.**Resources**(*client*)
Manage Orchestrator deployment resources.

list(*uuid*)
List resources for a deployment.

Parameters **uuid** (*str*) – The UUID of the deployment get the resources.

Returns A list of orpy.client.base.Resource

Return type list

show(*deployment_uuid*, *resource_uuid*)
Show details about a resource on a deployment.

Parameters

- **resource_uuid** (*str*) – The UUID of the deployment get the resource.
- **deployment_uuid** (*str*) – The UUID of the resource.

Returns The resource requested

Return type *orpy.client.base.Resource*

3.4 Information interface

class orpy.client.info.**Info**(*client*)
Get information about the Orchestrator.

get()
Get information about the Orchestrator.

Returns Information about the orchestrator.

Return type *orpy.client.base.OrchestratorInfo*

3.5 Orchestrator resources objects

class orpy.client.base.**BaseObject**(*info*)
Base class for all objects that represents orchestrator resoruces.

set_info(*key*, *value*)
Set an objects information with key, value.

Parameters

- **key** – the element to set
- **value** – the value for the element

to_dict()
Translate the object into a dictionary.

Returns A dictionary contaning the object representation

Return type dict

class orpy.client.base.**Deployment**(*info*)
Object that represents a deployment.

set_info(*key*, *value*)
Set an objects information with key, value.

Parameters

- **key** – the element to set
- **value** – the value for the element

to_dict()
Translate the object into a dictionary.

Returns A dictionary contaning the object representation

Return type dict

class orpy.client.base.**OrchestratorConfiguration**(*info*)
Object that represents the Orchestrtor information.

set_info(*key*, *value*)
Set an objects information with key, value.

Parameters

- **key** – the element to set
- **value** – the value for the element

to_dict()
Translate the object into a dictionary.

Returns A dictionary contaning the object representation

Return type dict

class orpy.client.base.**OrchestratorInfo**(*info*)
Object that represents the Orchestrtor information.

set_info(*key*, *value*)
Set an objects information with key, value.

Parameters

- **key** – the element to set
- **value** – the value for the element

to_dict()
Translate the object into a dictionary.

Returns A dictionary contaning the object representation

Return type dict

class orpy.client.base.**Resource**(*info*)
Object that represents a Resource.

set_info(*key*, *value*)
Set an objects information with key, value.

Parameters

- **key** – the element to set
- **value** – the value for the element

to_dict()
Translate the object into a dictionary.

Returns A dictionary contaning the object representation

Return type dict

class orpy.client.base.**TOSCATemplate**(*info*)
Object that repesents a TOSCA template.

set_info(*key*, *value*)
Set an objects information with key, value.

Parameters

- **key** – the element to set
- **value** – the value for the element

to_dict()
Translate the object into a dictionary.

Returns A dictionary contaning the object representation

Return type dict

CHAPTER 4

Additional Notes

4.1 Contributing

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

4.1.1 Types of Contributions

You can contribute in many ways:

Report Bugs

Report bugs at replace with the project page/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.
- If you can, provide detailed steps to reproduce the bug.
- If you don't have steps to reproduce the bug, just note your observations in as much detail as you can. Questions to start a discussion about the issue are welcome.

Fix Bugs

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

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "please-help" is open to whoever wants to implement it.

Please do not combine multiple feature enhancements into a single pull request.

Note: this project is very conservative, so new features that aren't tagged with "please-help" might not get into core. We're trying to keep the code base small, extensible, and streamlined. Whenever possible, it's best to try and implement feature ideas as separate projects outside of the core codebase.

Write Documentation

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

If you want to review your changes on the documentation locally, you can do::

```
pip install -r docs/requirements.txt
make servedocs
```

This will compile the documentation, open it in your browser and start watching the files for changes, recompiling as you save.

Submit Feedback

The best way to send feedback is to file an issue at replace with the project page/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 this is a volunteer-driven project, and that contributions are welcome :)

4.1.2 Setting Up the Code for Local Development

Here's how to set up `cookiecutter` for local development.

1. Fork the `cookiecutter` repo on GitHub.
2. Clone your fork locally::

   $ git clone git@github.com:your_name_here/cookiecutter.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::

   $ mkvirtualenv cookiecutter $ cd cookiecutter/ $ python setup.py develop
4. Create a branch for local development::

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

Now you can make your changes locally.

1. When you're done making changes, check that your changes pass the tests and flake8::

   $ pip install tox $ tox

Please note that tox runs flake8 automatically, since we have a test environment for it.

If you feel like running only the flake8 environment, please use the following command::

```
$ tox -e flake8
```

1. 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

2. Check that the test coverage hasn't dropped::

   $ tox -e cov-report

3. Submit a pull request through the GitHub website.

4.1.3 Contributor Guidelines

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 2.7, 3.3, 3.4, 3.5, 3.6, and PyPy on Appveyor and Travis CI.
4. Check https://travis-ci.org/audreyr/cookiecutter/pull_requests and https://ci.appveyor.com/project/audreyr/cookiecutter/history to ensure the tests pass for all supported Python versions and platforms.

Coding Standards

- PEP8
- Functions over classes except in tests
- Quotes via http://stackoverflow.com/a/56190/5549
  - Use double quotes around strings that are used for interpolation or that are natural language messages
  - Use single quotes for small symbol-like strings (but break the rules if the strings contain quotes)
  - Use triple double quotes for docstrings and raw string literals for regular expressions even if they aren't needed.
  - Example:

    .. code-block:: python

```
LIGHT_MESSAGES = {
    'English': "There are %(number_of_lights)s lights.",
    'Pirate':  "Arr! Thar be %(number_of_lights)s lights."
}

def lights_message(language, number_of_lights):
    """Return a language-appropriate string reporting the light count."""
    return LIGHT_MESSAGES[language] % locals()
```

(continues on next page)

(continued from previous page)

```
def is_pirate(message):
    """Return True if the given message sounds piratical."""
    return re.search(r"(?i)(arr|avast|yohoho)!", message) is not None
```

- Write new code in Python 3.

4.1.4 Testing with tox

Tox uses py.test under the hood, hence it supports the same syntax for selecting tests.

For further information please consult the `pytest usage docs_`.

To run a particular test class with tox::

```
$ tox -e py '-k TestFindHooks'
```

To run some tests with names matching a string expression::

```
$ tox -e py '-k generate'
```

Will run all tests matching "generate", test_generate_files for example.

To run just one method::

```
$ tox -e py '-k "TestFindHooks and test_find_hook"'
```

To run all tests using various versions of python in virtualenvs defined in tox.ini, just run tox.::

```
$ tox
```

This configuration file setup the pytest-cov plugin and it is an additional dependency. It generate a coverage report after the tests.

It is possible to tests with some versions of python, to do this the command is::

```
$ tox -e py27,py34,pypy
```

Will run py.test with the python2.7, python3.4 and pypy interpreters, for example.

4.2 Core Committer Guide

4.2.1 Vision and Scope

Core committers, use this section to:

- Guide your instinct and decisions as a core committer
- Limit the codebase from growing infinitely

Command-Line Accessible

```
* Provides a command-line utility that creates projects from cookiecutters
* Extremely easy to use without having to think too hard
* Flexible for more complex use via optional arguments
```

(continues on next page)

(continued from previous page)

```
API Accessible
~~~~~~~~~~~~~~

* Entirely function-based and stateless (Class-free by intentional design)
* Usable in pieces for developers of template generation tools

Being Jinja2-specific
~~~~~~~~~~~~~~~~~~~~~

* Sets a standard baseline for project template creators, facilitating reuse
* Minimizes the learning curve for those who already use Flask or Django
* Minimizes scope of Cookiecutter codebase

Extensible
~~~~~~~~~~

Being extendable by people with different ideas for Jinja2-based project template␣
↪tools.

* Entirely function-based
* Aim for statelessness
* Lets anyone write more opinionated tools

Freedom for Cookiecutter users to build and extend.

* No officially-maintained cookiecutter templates, only ones by individuals
* Commercial project-friendly licensing, allowing for private cookiecutters and␣
↪private Cookiecutter-based tools

Fast and Focused
~~~~~~~~~~~~~~~~

Cookiecutter is designed to do one thing, and do that one thing very well.

* Cover the use cases that the core committers need, and as little as possible beyond␣
↪that :)
* Generates project templates from the command-line or API, nothing more
* Minimize internal line of code (LOC) count
* Ultra-fast project generation for high performance downstream tools

Inclusive
~~~~~~~~~

* Cross-platform and cross-version support are more important than features/
↪functionality
* Fixing Windows bugs even if it's a pain, to allow for use by more beginner coders

Stable
~~~~~~

* Aim for 100% test coverage and covering corner cases
* No pull requests will be accepted that drop test coverage on any platform,␣
↪including Windows
* Conservative decisions patterned after CPython's conservative decisions with␣
↪stability in mind
* Stable APIs that tool builders can rely on
```

(continues on next page)

(continued from previous page)

```
* New features require a +1 from 3 core committers

VCS-Hosted Templates
~~~~~~~~~~~~~~~~~~~

Cookiecutter project templates are intentionally hosted VCS repos as-is.

* They are easily forkable
* It's easy for users to browse forks and files
* They are searchable via standard Github/Bitbucket/other search interface
* Minimizes the need for packaging-related cruft files
* Easy to create a public project template and host it for free
* Easy to collaborate

Process: Pull Requests
----------------------

If a pull request is untriaged:

* Look at the roadmap
* Set it for the milestone where it makes the most sense
* Add it to the roadmap

How to prioritize pull requests, from most to least important:

#. Fixes for broken tests. Broken means broken on any supported platform or Python␣
↪version.
#. Extra tests to cover corner cases.
#. Minor edits to docs.
#. Bug fixes.
#. Major edits to docs.
#. Features.

Ensure that each pull request meets all requirements in this checklist:
https://gist.github.com/audreyr/4feef90445b9680475f2

Process: Issues
---------------

If an issue is a bug that needs an urgent fix, mark it for the next patch release.
Then either fix it or mark as please-help.

For other issues: encourage friendly discussion, moderate debate, offer your thoughts.

New features require a +1 from 2 other core committers (besides yourself).

Process: Roadmap
----------------

The roadmap is https://github.com/audreyr/cookiecutter/milestones?direction=desc&
↪sort=due_date&state=open

Due dates are flexible. Core committers can change them as needed. Note that GitHub␣
↪sort on them is buggy.

How to number milestones:
```

(continues on next page)

(continued from previous page)

```
* Follow semantic versioning. See http://semver.org

Milestone size:

* If a milestone contains too much, move some to the next milestone.
* Err on the side of more frequent patch releases.

Process: Pull Request merging and HISTORY.rst maintenance
-----------------------------------------------------------

If you merge a pull request, you're responsible for updating `AUTHORS.rst` and␣
↪`HISTORY.rst`

When you're processing the first change after a release, create boilerplate following␣
↪the existing pattern:

.. code-block:: rest

    x.y.z (Development)
    ~~~~~~~~~~~~~~~~~~~

    The goals of this release are TODO: release summary of features

    Features:

    * Feature description, thanks to @contributor (#PR).

    Bug Fixes:

    * Bug fix description, thanks to @contributor (#PR).

    Other changes:

    * Description of the change, thanks to @contributor (#PR).

    .. _`@contributor`: https://github.com/contributor

Process: Accepting Template Pull Requests
-----------------------------------------

#. Run the template to generate the project.
#. Attempt to start/use the rendered project.
#. Merge the template in.
#. Update the history file.

.. note:: Adding a template doesn't give authors credit.


Process: Generating CONTRIBUTING.rst
------------------------------------

From the `cookiecutter` project root::

    $ make contributing

This will generate the following message::
```

(continues on next page)

(continued from previous page)

```
    rm CONTRIBUTING.rst
    touch CONTRIBUTING.rst
    cat docs/contributing.rst >> CONTRIBUTING.rst
    echo "\n\n" >> CONTRIBUTING.rst
    cat docs/types_of_contributions.rst >> CONTRIBUTING.rst
    echo "\n\n" >> CONTRIBUTING.rst
    cat docs/contributor_setup.rst >> CONTRIBUTING.rst
    echo "\n\n" >> CONTRIBUTING.rst
    cat docs/contributor_guidelines.rst >> CONTRIBUTING.rst
    echo "\n\n" >> CONTRIBUTING.rst
    cat docs/contributor_testing.rst >> CONTRIBUTING.rst
    echo "\n\n" >> CONTRIBUTING.rst
    cat docs/core_committer_guide.rst >> CONTRIBUTING.rst
    echo "\n\nAutogenerated from the docs via \`make contributing\`" >> CONTRIBUTING.
↪rst
    echo "WARNING: Don't forget to replace any :ref: statements with literal names"
    WARNING: Don't forget to replace any :ref: statements with literal names

Process: Your own code changes
------------------------------

All code changes, regardless of who does them, need to be reviewed and merged by␣
↪someone else.
This rule applies to all the core committers.

Exceptions:

* Minor corrections and fixes to pull requests submitted by others.
* While making a formal release, the release manager can make necessary, appropriate␣
↪changes.
* Small documentation changes that reinforce existing subject matter. Most commonly␣
↪being, but not limited to spelling and grammar corrections.

Responsibilities
----------------

#. Ensure cross-platform compatibility for every change that's accepted. Windows, Mac,
↪ Debian & Ubuntu Linux.
#. Ensure that code that goes into core meets all requirements in this checklist:␣
↪https://gist.github.com/audreyr/4feef90445b9680475f2
#. Create issues for any major changes and enhancements that you wish to make.␣
↪Discuss things transparently and get community feedback.
#. Don't add any classes to the codebase unless absolutely needed. Err on the side of␣
↪using functions.
#. Keep feature versions as small as possible, preferably one new feature per version.
#. Be welcoming to newcomers and encourage diverse new contributors from all␣
↪backgrounds. See the Python Community Code of Conduct (https://www.python.org/psf/
↪codeofconduct/).

Becoming a Core Committer
-------------------------

Contributors may be given core commit privileges. Preference will be given to those␣
↪with:

A. Past contributions to Cookiecutter and other open-source projects. Contributions␣
↪to Cookiecutter include both code (both accepted and pending) and friendly␣
↪participation in the issue tracker. Quantity and quality are considered.
```

(continues on next page)

(continued from previous page)

```
B. A coding style that the other core committers find simple, minimal, and clean.
C. Access to resources for cross-platform development and testing.
D. Time to devote to the project regularly.


Autogenerated from the docs via `make contributing`
```

4.3 Contributor Covenant Code of Conduct

4.3.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

4.3.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

4.3.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

4.3.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

4.3.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at [INSERT EMAIL ADDRESS]. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

4.3.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

4.4 Orpy Release Notes

4.4.1 Current (un)Release Notes

1.0.0

New Features

- New command *configuration show* allows to retrieve information from the orchestrator regarding its configuration.
- New *–long* option to provide long output when listing the deployments.

Other Notes

- Output has changed in order to be more human readable.

0.5.0

New Features

- Leverage oidc-agent to get the user's access token when interacting with the INDIGO PaaS orchestrator.

0.3.0

New Features

- This version of orpy stops using bare dictionaries, and relies on objects for each of the REST resources.

0.2.0

New Features

- Initial version of the orchestrator API bindings and CLI with full functionality. Implements the complete orchestrator REST API.

Security Issues

- Authentication only possible through the usage of an OpenID Connect access token.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index

Symbols

B

C

D

G

H

I

L

O

P

R

S

T

U