

---

# **python-copr Documentation**

*Release 1.63*

**Valentin Gologuzov**

**Feb 05, 2020**



---

## Contents

---

|          |                            |           |
|----------|----------------------------|-----------|
| <b>1</b> | <b>Installation</b>        | <b>3</b>  |
| <b>2</b> | <b>Contact</b>             | <b>5</b>  |
| <b>3</b> | <b>Usage</b>               | <b>7</b>  |
| <b>4</b> | <b>Indices and tables</b>  | <b>45</b> |
|          | <b>Python Module Index</b> | <b>47</b> |
|          | <b>Index</b>               | <b>49</b> |



Python-copr is a python client to access the Copr build service through its [API](#).

Python-copr right now is in alpha stage, so expect lot of changes. Now it targets python 2.6+ and python3.3+.



# CHAPTER 1

---

## Installation

---

### 1.1 Dependencies:

```
python2.6+
python-requests
python-six
```

### 1.2 repo

Available for fedora 21+

```
dnf install python-copr python-copr-doc
```

### 1.3 source

```
git clone https://pagure.io/copr/copr.git
cd python
# enable virtualenv if needed
pip install -r requirements.txt
python setup.py install
```





## CHAPTER 2

---

### Contact

---

If you have any questions, please contact us:

- IRC: [#fedora-buildsys@irc.freenode.net](irc://irc.freenode.net/#fedora-buildsys)
- mailing list: [copr-devel@lists.fedorahosted.org](mailto:copr-devel@lists.fedorahosted.org) [signup] [archives]



Copr currently supports three independent API versions.

## 3.1 Legacy client

**Warning:** Legacy client is obsolete, please use Client version 3 instead. This document describes the migration process.

All interaction are done through `copr.CoprClient`. It can be created directly or using config file `/etc/copr.conf`. `CoprClient` offers methods that directly reflect Copr api. Received data are wrapped into the `Response` object. Depending on used methods Responses will have different set of provided attributes and methods.

### 3.1.1 Create client

```
from copr.client import CoprClient

# using default config location ~/.config/copr
cl = CoprClient.create_from_file_config()

# using config at ~/.config/alt_copr
cl = CoprClient.create_from_file_config("~/.config/alt_copr")

# directly
cl = CoprClient(username="foo", login="abcd", token="efgk", copr_url="http://example.
↳com/")
```

### 3.1.2 Get projects list

```
# your projects
result = cl.get_projects_list().projects_list

# other user projects
result = cl.get_projects_list(username="rhscl")

# print list for humans and machines
from pprint import pprint
for prj in result.projects_list:
    print(prj)
    pprint(prj.get_project_details().data)
```

### 3.1.3 Search for projects

```
result = cl.search_projects("python")
# print list for humans
for prj in result.projects_list:
    print(prj)
```

### 3.1.4 Create/update/delete project

```
result = cl.create_project("hello_world", chroots=["fedora-20-x86_64"],
    description="My cool app")

# assert correct description
assert result.handle.get_project_details().description == "My cool app"

# add instruction
result.handle.modify_project(instructions="How does one patch KDE2 under FreeBSD?")
# which is shorter than
cl.modify_project("hello_world", instructions="...")

# delete project
result.handle.delete_project()
# again shortcut for
cl.delete_project("hello_world")
```

### 3.1.5 Work with builds

```
# building new package
result = cl.create_new_build("hello_world",
    pkgs=["http://example.com/pkg.src.rpm",])

# retrieve build statuses
for bw in result.builds_list:
    print("{0}:{1}".format(bw.build_id, bw.handle.get_build_details().status))

# cancel all created build
for bw in result.builds_list:
```

(continues on next page)

(continued from previous page)

```

bw.handle.cancel_build()

# get build status for each chroot
for bw in result.builds_list:
    print("build: {0}".format(bw.build_id))
    for ch, status in bw.handle.get_build_details().data["chroots"].items():
        print("\t chroot {0}:\t {1}".format(ch, status))

# simple build progress:
import time, datetime
watched = set(result.builds_list)
done = set()
while watched != done:
    print("time: {0}".format(datetime.datetime.now()))
    for bw in watched:
        if bw in done:
            continue
        status = bw.handle.get_build_details().status
        print("{0}: {1}".format(bw.build_id, status))
        if status in ["skipped", "failed", "succeeded"]:
            done.add(bw)
    time.sleep(10)

# cancel all created build
for bw in result.builds_list:
    bw.handle.cancel_build()

```

See method signatures and response objects in the auto generated documentation:

### 3.1.6 CoprClient

**class** `copr.client.client.CoprClient` (*username=None, login=None, token=None, copr\_url=None, no\_config=False*)

Main interface to the copr service

#### Variables

- **username** (*unicode*) – username used by default for all requests
- **login** (*unicode*) – user login, used for identification
- **token** (*unicode*) – copr api token
- **copr\_url** (*unicode*) – used as copr projects root

#### Could be created:

- directly
- using static method `CoprClient.create_from_file_config()`

**static** `create_from_file_config` (*filepath=None, ignore\_error=False*)  
Creates Copr client using the information from the config file.

#### Parameters

- **filepath** (*str*) – specifies config location, default: “~/config/copr”
- **ignore\_error** (*bool*) – When true creates default Client without credentials

**Return type** *CoprClient*

**get\_build\_details** (*build\_id*, *projectname=None*, *username=None*)

Returns build details.

**Parameters**

- **build\_id** (*int*) – Build identifier
- **projectname** – [optional] Copr project name
- **username** – [optional] Copr project owner

**Returns**

*CoprResponse* with additional fields:

- **handle**: *BuildHandle*
- text fields: “project”, “owner”, “status”, “results”, “submitted\_on”, “started\_on”, “ended\_on”, “built\_pkgs”, “src\_pkg”, “src\_version”

**cancel\_build** (*build\_id*, *projectname=None*, *username=None*)

Cancels build. Auth required. If build can't be canceled do nothing.

**Parameters**

- **build\_id** (*int*) – Build identifier
- **projectname** – [optional] Copr project name
- **username** – [optional] Copr project owner

**Returns**

*CoprResponse* with additional fields:

- **handle**: *BuildHandle*
- text fields: “status”

**create\_new\_build** (*projectname*, *pkgs*, *username=None*, *timeout=None*, *memory=None*, *chroots=None*, *background=False*, *progress\_callback=None*)

Creates new build

**Parameters**

- **projectname** – name of Copr project (without user namespace)
- **pkgs** – list of packages to include in build
- **username** – [optional] use alternative username
- **timeout** – [optional] build timeout
- **memory** – [optional] amount of required memory for build process
- **chroots** – [optional] build only with given chroots
- **background** – [optional] mark the build as a background job.
- **progress\_callback** – [optional] a function that received a

MultipartEncoderMonitor instance for each chunk of uploaded data

**Returns**

*CoprResponse* with additional fields:

- **builds\_list**: list of *BuildWrapper*

**get\_project\_details** (*projectname*, *username=None*)

Returns project details

#### Parameters

- **projectname** – Copr projectname
- **username** – [optional] use alternative username

#### Returns

*CoprResponse* with additional fields:

- text fields: “description”, “instructions”, “last\_modified”, “name”
- **chroots**: list of *ProjectChrootWrapper*

**delete\_project** (*projectname*, *username=None*)

Deletes the entire project. Auth required.

#### Parameters

- **projectname** – Copr projectname
- **username** – [optional] use alternative username

#### Returns

*CoprResponse* with additional fields:

- text fields: “message”

**create\_project** (*username*, *projectname*, *chroots*, *description=None*, *instructions=None*, *repos=None*, *initial\_pkgs=None*, *disable\_createrepo=None*, *unlisted\_on\_hp=False*, *enable\_net=True*, *persistent=False*, *auto\_prune=True*, *use\_bootstrap\_container=None*)

Creates a new copr project Auth required.

#### Parameters

- **projectname** – User or group name
- **projectname** – Copr project name
- **chroots** – List of target chroots
- **description** – [optional] Project description
- **instructions** – [optional] Instructions for end users
- **disable\_createrepo** – [optional] disables automatic repo meta-data regeneration, “true”/“false” string
- **unlisted\_on\_hp** – [optional] Project will not be shown on COPR HP
- **enable\_net** – [optional] If builder can access net for builds in this project
- **persistent** – [optional] If builds and the project are undeletable
- **auto\_prune** – [optional] If backend auto-deletion script should be run for the project
- **use\_bootstrap\_container** – [optional] If mock bootstrap container is used to initialize the buildroot

#### Returns

*CoprResponse* with additional fields:

- **handle:** *ProjectHandle*
- text fields: “message”

**modify\_project** (*projectname*, *username=None*, *description=None*, *instructions=None*, *repos=None*, *disable\_createrepo=None*, *unlisted\_on\_hp=None*, *enable\_net=None*, *auto\_prune=None*, *use\_bootstrap\_container=None*, *chroots=None*)

Modifies main project configuration. Auth required.

#### Parameters

- **projectname** – Copr project name
- **username** – [optional] use alternative username
- **description** – [optional] project description
- **instructions** – [optional] instructions for end users
- **repos** – [optional] list of additional repos to be used during the build process
- **repos** – [optional] list of additional repos to be used during
- **disable\_createrepo** – [optional] disables automatic repo meta-data regeneration
- **unlisted\_on\_hp** – [optional] Project will not be shown on COPR HP
- **enable\_net** – [optional] If builder can access net for builds in this project
- **auto\_prune** – [optional] If backend auto-deletion script should be run for the project
- **use\_bootstrap\_container** – [optional] If mock bootstrap container is used to initialize the buildroot
- **chroots** – [optional] list of chroots that should be enabled in the project. When not *None*, selected chroots will be enabled while current chroots will not remain enabled if they are not specified.

#### Returns

*CoprResponse* with additional fields:

- **handle:** *ProjectHandle*
- text fields: “buildroot\_pkgs”

**get\_projects\_list** (*username=None*)

Returns list of projects created by the user

**Parameters** **username** – [optional] use alternative username

#### Returns

*CoprResponse* with additional fields:

- **projects\_list:** list of *ProjectWrapper*

**get\_project\_chroot\_details** (*projectname*, *chrootname*, *username=None*)

Returns details of chroot used in project

#### Parameters

- **projectname** – Copr project name
- **chrootname** – chroot name
- **username** – [optional] use alternative username



**Returns**

*CoprResponse* with additional fields:

- **handle:** *ProjectChrootHandle*
- text fields: “buildroot\_pkgs”

**modify\_project\_chroot\_details** (*projectname, chrootname, pkgs=None, username=None*)  
 @deprecated to edit\_chroot Modifies chroot used in project

**Parameters**

- **projectname** – Copr project name
- **chrootname** – chroot name
- **username** – [optional] use alternative username

**Returns**

*CoprResponse* with additional fields:

- **handle:** *ProjectChrootHandle*
- text fields: “buildroot\_pkgs”

**search\_projects** (*query*)  
 Search projects by substring

**Parameters** **query** – substring to search

**Returns**

*CoprResponse* with additional fields:

- **projects\_list:** list of *ProjectWrapper*

### 3.1.7 Responses

**class** `copr.client.responses.CoprResponse` (*client, method, data, request\_kwargs=None, parsers=None*)

Wrapper for Copr api responses

**Variables**

- **handle** – handle object which provide shortcuts based on request and/or response data (*BaseHandle* and its derivatives)
- **data** (*dict*) – json structure from Copr api

**class** `copr.client.responses.BaseHandle` (*client, username=None, response=None, \*\*kwargs*)  
 Handles provide convenient shortcut methods. Useful methods provided by derived classes.

Example:

```
response = client.create_project("copr")
response.handle # <-- ProjectHandle object
print(response.handle.get_project_details().data)
```

**class** `copr.client.responses.ProjectHandle` (*client, projectname, \*args, \*\*kwargs*)  
 Handle to deal with a single Copr project

**get\_project\_details** ()  
 Shortcut to `get_project_details()`

**modify\_project** (*\*\*kwargs*)  
Shortcut to *modify\_project()*

**delete\_project** ()  
Shortcut to *delete\_project()*

**create\_new\_build** (*src\_pkgs, chroots=None*)  
Shortcut to `:meth:`~.client.CoprClient.create_new_build``

**class** `copr.client.responses.BuildHandle` (*client, build\_id, \*args, \*\*kwargs*)  
Handle to deal with a single build

**project\_handle**  
Shortcut for *responses.ProjectHandle*

**get\_build\_details** ()  
Shortcut to *get\_build\_details()*

**cancel\_build** ()  
Shortcut to *cancel\_build()*

**class** `copr.client.responses.ProjectChrootHandle` (*client, chrootname, \*args, \*\*kwargs*)  
Handle to deal with a single project chroot

**get\_project\_chroot\_details** ()  
Shortcut to *get\_project\_chroot\_details()*

**modify\_project\_chroot\_details** (*pkgs=None*)  
Shortcut to *modify\_project\_chroot\_details()*

### 3.1.8 Data wrappers

**class** `copr.client.responses.ProjectWrapper` (*client, username, projectname, description=None, instructions=None, yum\_repos=None, additional\_repos=None*)

Helper class to represent project objects

`__str__` overridden to produces pretty formatted representation

#### Variables

- **handle** – *responses.ProjectHandle*
- **username** – project owner
- **projectname** – project names

**class** `copr.client.responses.BuildWrapper` (*client, username, projectname, build\_id, status=None*)

Helper class to represent build objects

#### Variables

- **handle** – *responses.BuildHandle*
- **username** – project owner
- **projectname** – project names
- **build\_id** (*int*) – build identifier

**class** `copr.client.responses.ProjectChrootWrapper` (*client, username, projectname, chrootname, repo\_url=None*)

Helper class to represent project chroot objects

### Variables

- **handle** – `responses.ProjectChrootHandle`
- **username** – project owner
- **projectname** – project names
- **chrootname** – chroot name

## 3.2 Client version 2

**Warning:** Client version 2 is obsolete, please use Client version 3 instead.

New package `copr.ClientV2` supports APIv2 and should eventually replace older one. New API and `Client_v2` provides more simple and uniform approach for communication with the Copr service.

Client mostly reflects resource based API nature. Client object offers range of methods to query from service. Response objects contains requested information and also provides helper methods to execute new requests in the context of retrieved objects.

### 3.2.1 Client initialization

To create client instance use either of two functions:

- `copr.create_client2_from_params()`
- `copr.create_client2_from_file_config()`

Config file should be generated by the Copr service itself ([Api info page](#)). Don't forget to login.

### 3.2.2 Resources

Client represents API with two kinds of resources: Individuals and Collections. For example when we request all projects with name `copr` we would receive collection `ProjectList` resource:

```
>>> from copr import create_client2_from_params
# using dev server for test
>>> cl = create_client2_from_params(root_url="http://copr-fe-dev.cloud.
↳ fedoraproject.org/")

>>> projects = cl.projects.get_list(name="copr", limit=3)
>>> for p in projects:
>>>     print(p)
<Project #1: msuchy/copr>
<Project #1503: vgologuz/copr>
<Project #2796: mosquito/copr>
```

Access to elements in collection is done through iterator interface. Since API limits number of elements retrieved in the one request, collections has method `next_page()` to retrieve more objects:

```
>>> more_projects = projects.next_page()
>>> for p in more_projects:
>>>     print(p)
<Project #2805: esmil/copr>
<Project #4266: frostyx/copr>
```

If we already knew project id we could get an individual *Project* resource:

```
>>> p = cl.projects.get_one(1835)
```

Individual resource allows to directly access entity properties and also provides some helper functions:

```
>>> print(p.owner, p.name)
(u'saltstack', u'salt')
# obtain active build chroots
>>> print("\n".join(map(str, p.get_project_chroot_list())))
<Project chroot: fedora-22-x86_64, additional packages: [], comps size if_
↪any: 0>
<Project chroot: fedora-22-i386, additional packages: [], comps size if any:_
↪0>
# change project description (require auth)
>>> p.description = u"Hello world!"
>>> p.update()
# instead of cl.projects.update(p._entity)
```

### 3.2.3 Error handling

### 3.2.4 Resources info

#### Project

Project resource represents copr projects and operations with them.

Access to the projects is done through *projects()* property of initialized *CoprClient*. That property is an instance of *ProjectHandle*. Projects are represented by *Project* class.

## Project entity attributes

| Field              | Type                 | Can edit? | Description  |
|--------------------|----------------------|-----------|--|
| id                 | number               | no        | unique identifier  |
| owner              | string               | no        | username of the project owner  |
| group              | string               | no        | <p><b>name of the group which owns the project,</b></p> <ul style="list-style-type: none"> <li>• MAY be specified during a project creation to create a group managed project</li> </ul> |
| name               | string               | no        | <p><b>name of the project</b></p> <ul style="list-style-type: none"> <li>• MUST be specified during a project creation</li> </ul>  |
| description        | string               | yes       | project description  |
| instructions       | string               | yes       | installation instructions  |
| homepage           | string(URL)          | yes       | project homepage URL   |
| contact            | string(URL or email) | yes       | contact with the project maintainer  |
| disable_createrepo | bool                 | yes       | disables automatic repository metadata generation  |
| build_enable_net   | bool                 | yes       | set default value for new builds option <code>enable_net</code>  |
| repos              | list of string       | yes       | list of additional repositories to be enabled during the build   |

**Note:** all following examples assume that we use `cl` as an instance of `client_v2.client.CoprClient`

### Get projects list

```
>>> plist_1 = cl.project.get_list(limit=10)
# filter by name
>>> plist_2 = cl.project.get_list(name="copr")
# search by string
>>> plist_2 = cl.project.get_list(search_query="copr")
```

## Get one project

```
>>> p = cl.projects.get_one(1835)
```

## Modify project parameters

```
>>> p.description = "Nothing"
>>> p.update()
```

## Delete project

```
>>> p.delete()
```

## Create new project

---

**Note:** Here you could also provide list of chroots, which should be activated. Use key `chroots`.

---

```
>>> res = cl.projects.create(name="my_cool_project",
                             owner="vgologuz",
                             instructions="don't touch me!",
                             chroots=["fedora-22-x86_64"])
>>> print(res)
<Project #5384: vgologuz/my_cool_project>
```

## Access project chroots

---

**Note:** see also *Project chroot*

---

```
# get all lists
>>> chroots = p.get_project_chroot_list()
>>> print("\n".join(map(str, chroots)))
<Project chroot: fedora-21-x86_64, additional packages: [], comps size if any: 0>
<Project chroot: fedora-21-i386, additional packages: [], comps size if any: 0>
# get one chroot
>>> chroot_1 = p.get_project_chroot("fedora-22-i386")
# enable chroot for project
>>> p.enable_project_chroot("fedora-22-x86_64")
```

## Access project builds

---

**Note:** see also *Build*

---

```

>>> p.get_builds(limit=5)
>>> pbuilds = p.get_builds(limit=5)
>>> print(pbuilds[3])
<Build #138414 state: failed>

# submit new builds
>>> p.create_build_from_url(srpm_url="http://example.com/my.src.rpm")
>>> p.create_build_from_file(file_path="/tmp/my.src.rpm")

```

## Project chroot

Projects Chroots allows to enable and disable target chroots and modify project settings dedicated for specific chroots.

Access to the project chroots is done through `project_chroots()` property of initialized `CoprClient`. That property is an instance of `ProjectChrootHandle`.

However it's usually more convenient to access project chroots from an instance of `Project` using methods `get_project_chroot_list()` or `get_project_chroot()`.

Chroot are represented by `ProjectChroot` class.

## Project chroot entity attributes

| Field          | Type            | Can edit? | Description                                 |
|----------------|-----------------|-----------|---|
| name           | string          | no        | chroot name                                 |
| buildroot_pkgs | list of strings | yes       | packages to be installed into the buildroot |
| comps          | string          | yes       | content of the <code>comps.xml</code>       |
| comps_name     | string          | yes       | name of the uploaded comps file             |
| comps_len      | int             | no        | size of the uploaded comps file (bytes)     |

---

**Note:** all following examples assume that we use `cl` as an instance of `client_v2.client.CoprClient` and `p` as an instance of `Project`

---

## Get project chroots list

```

>>> pc_list = cl.project_chroots.get_list(project=p)
# or more simple
>>> pc_list = p.get_project_chroot_list()
>>> map(str, pc_list)
['<Project chroot: fedora-21-x86_64, additional packages: [], comps size if any: 0>',
 '<Project chroot: epel-7-x86_64, additional packages: [], comps size if any: 0>']

```

## Get one project chroot

```

>>> pc = cl.project_chroots.get_one(project=p, name="fedora-23-x86_64")
# or
>>> pc = p.get_project_chroot("fedora-23-x86_64")

```

(continues on next page)

(continued from previous page)

```
>>> print(pc)
<Project chroot: fedora-23-x86_64, additional packages: [], comps size if any: 0>
```

### Modify project chroot

```
>>> pc.buildroot_pkgs = ["scl-utils",]
>>> pc.update()
```

### Disable project chroot

```
>>> pc.disable()
```

## Build

Build resource allows to submit new builds and access current build progress. In fact, build consists of a few tasks, one per chroot, and detailed information is available through *Build task*.

Access to the builds is done through *builds()* property of initialized *CoprClient*. That property is an instance of *BuildHandle*.

It may be more convenient to access builds in context of a project using method *get\_builds()*.

Builds are represented by *Build* class.

### Build entity attributes

| Field           | Type              | Description  |
|-----------------|-------------------|--|
| id              | int               | unique build identifier  |
| state           | string            | current state of the build, value is aggregated from build tasks                       |
| submitted_on    | int(unixtime UTC) | time of the build submission   |
| started_on      | int(unixtime UTC) | time when the first build task started, otherwise null                                 |
| ended_on        | int(unixtime UTC) | time when the last build task ended, otherwise null                                    |
| source_type     | string            | method used for build creation   |
| source_metadata | json object       | build source information   |
| package_version | string            | version of the source package  |
| package_name    | string            | name of the source package   |
| enable_net      | bool              | defines if network is available during the build                                       |
| repos           | list of string    | list of additional repositories enabled during the build                               |
| built_packages  | list of hash maps | list of the built packages, each hash map has two keys: <i>name</i> and <i>version</i> |
| submitter       | string            | name of the user who submitted the build   |



---

**Note:** Only the `state` field is editable by the PUT method. All other fields are read-only.

---

### Get builds list

```
>>> blist_1 = cl.builds.get_list(limit=2)
>>> print(map(str, blist_1))
['<Build #138426 state: succeeded>', '<Build #138425 state: succeeded>']
# using the project object
>>> blist_2 = p.get_builds(limit=2)
>>> print(map(str, blist_2))
['<Build #138423 state: failed>', '<Build #138421 state: failed>']
```

### Get one build

```
>>> b = cl.builds.get_one(138421)
>>> print(b)
<Build #138421 state: failed>
```

### Create new build

```
# using the url to srpm
>>> b1 = cl.builds.create_from_url(project_id=3806, srpm_url="http://example.com/my.
↳src.rpm")
# or using project object
>>> b2 = p.create_build_from_url("http://example.com/my.src.rpm")
>>> print(map(str, [b1, b2]))
['<Build #138431 state: importing>', '<Build #138430 state: importing>']

# and using the file upload
>>> b3 = cl.builds.create_from_file(project_id=3806, file_path="/tmp/hello-2.8-1.fc20.
↳src.rpm")
>>> b4 = p.create_build_from_file("/tmp/hello-2.8-1.fc20.src.rpm")
```

### Cancel build

```
>>> b1.cancel()
```

### Delete build

```
>>> b1.delete()
```

### Build task

Build task represents information about individual build tasks. One task is responsible for one chroot.

Access to the build tasks is done through `build_tasks()`. property of initialized `CoprClient`. That property is an instance of `BuildTaskHandle`.

It may be more convenient to access build tasks in context of a build using method `get_build_tasks()` of access build tasks in context of a project using method `get_build_tasks()`

Build tasks are represented by `BuildTask` class.

### Build task entity attributes

| Field                       | Type                           | Description   |
|-----------------------------|--------------------------------|---|
| <code>chroot_name</code>    | <code>str</code>               | chroot name   |
| <code>build_id</code>       | <code>int</code>               | unique build identifier   |
| <code>state</code>          | <code>str</code>               | current build task state  |
| <code>started_on</code>     | <code>int(unixtime UTC)</code> | time when the build chroot started  |
| <code>ended_on</code>       | <code>int(unixtime UTC)</code> | time when the build chroot ended  |
| <code>git_hash</code>       | <code>str</code>               | hash of the <code>git</code> commit in <code>dist-git</code> used for the build |
| <code>result_dir_url</code> | <code>str(URL)</code>          | location of the build results   |

---

**Note:** Build Task doesn't currently support any modifications, so all fields are read-only.

---

---

**Note:** all following examples assume that we use `cl` as an instance of `client_v2.client.CoprClient`, `p` as an instance of `Project`, `b` as an instance of `Build`,

---

### Get build tasks list

```
>>> bt_list_1 = cl.build_tasks.get_list(state=BuildStateValues.FAILED, limit=5)
>>> map(str, bt_list_1)
['<Build task #17-fedora-18-x86_64, state: failed>',
 '<Build task #17-fedora-18-i386, state: failed>',
 '<Build task #17-fedora-19-x86_64, state: failed>',
 '<Build task #17-fedora-19-i386, state: failed>',
 '<Build task #107850-fedora-23-x86_64, state: failed>']

# using project object
>>> bt_list_2 = p.get_build_tasks(limit=5)
>>> map(str, bt_list_2)
['<Build task #86705-epel-6-i386, state: succeeded>',
 '<Build task #86705-epel-6-x86_64, state: failed>',
 '<Build task #86705-fedora-rawhide-x86_64, state: succeeded>',
 '<Build task #86705-fedora-rawhide-i386, state: failed>',
 '<Build task #86705-fedora-20-x86_64, state: succeeded>']

# using build object
>>> bt_list_3 = b.get_build_tasks(limit=5)
>>> map(str, bt_list_3)
['<Build task #87165-epel-6-i386, state: failed>',
 '<Build task #87165-epel-6-x86_64, state: failed>',
 '<Build task #87165-fedora-rawhide-x86_64, state: succeeded>',
```

(continues on next page)

(continued from previous page)

```
'<Build task #87165-fedora-rawhide-i386, state: succeeded>',
'<Build task #87165-fedora-20-x86_64, state: succeeded>']
```

## Get single build task

```
>>> bt = cl.build_tasks.get_one(106897, "epel-6-i386")
>>> print(bt.state, bt.result_dir_url)
(u'succeeded', u'http://copr-be-dev.cloud.fedoraproject.org/results/rineau/
↳libQGLViewer-qt5/epel-6-i386/libQGLViewer-2.5.1-5.fc21')
```

## Mock chroot

Mock chroot resources represents available chroots for builds. API provides only read-only access, since configuration of the build chroots is done by the service administrator.

Access to the mock chroots is done through `mock_chroots()` property of initialized `CoprClient`. That property is an instance of `MockChrootHandle`. Mock chroots are represented by `MockChroot` class.

## Mock chroot entity attributes

| Field      | Type | Description   |
|------------|------|---|
| name       | str  | chroot name   |
| os_release | str  | name of distribution system, e.g.: epel, fedora           |
| os_version | str  | version of distribution system, e.g.: 7, 22               |
| arch       | str  | architecture of distribution, e.g.: i386, x86_64, ppc64le |
| is_active  | bool | defines if this chroot is available for builds            |

---

**Note:** all following examples assume that we use `cl` as an instance of `client_v2.client.CoprClient`

---

## Get mock chroot list

```
>>> mlist = map(str, cl.mock_chroots.get_list(active_only=True))
>>> print("\n".join(map(str, mlist)))
<Mock chroot: epel-6-i386 is active: True>
<Mock chroot: epel-6-x86_64 is active: True>
<Mock chroot: epel-5-i386 is active: True>
...

```

## Get single mock chroot

```
>>> mc = cl.mock_chroots.get_one("fedora-22-i386")
>>> print(mc)
<Mock chroot: fedora-22-i386 is active: True>
```

(continues on next page)

```
>>> print(mc.name, mc.os_release, mc.os_version, mc.arch)
(u'fedora-22-i386', u'fedora', u'22', u'i386')
```

## 3.2.5 Autodoc

### General

`copr.create_client2_from_params` (*root\_url=None, login=None, token=None*)

Create client instance using the given parameters

#### Parameters

- **root\_url** (*str*) – Url to the Copr service, default: “http://copr.fedoraproject.org”
- **login** (*str*) – api login
- **token** (*str*) – api token

**Return type** `client_v2.client.CoprClient`

`copr.create_client2_from_file_config` (*filepath=None, ignore\_error=False*)

Creates Copr client using the information from the config file.

#### Parameters

- **filepath** (*str*) – specifies config location, default: “~/config/copr”
- **ignore\_error** (*bool*) – When true and config is missing, creates default Client without credentials

**Return type** `client_v2.client.CoprClient`

### CoprClient

**class** `copr.client_v2.client.CoprClient` (*net\_client, root\_url=None, no\_config=False*)

Main interface to the copr service

#### Parameters

- **net\_client** (*NetClient*) – wrapper for http requests
- **root\_url** (*unicode*) – used as copr projects root
- **no\_config** (*bool*) – helper flag to indicate that no config was provided

#### Could be created:

- using static method `create_from_file_config()`
- using static method `create_from_params()`

If you create Client directly call `CoprClient.post_init()` method after the creation.

#### projects

**Return type** `ProjectHandle`

#### project\_chroots

**Return type** `ProjectChrootHandle`

**builds**

**Return type** *BuildHandle*

**build\_tasks**

**Return type** *BuildTaskHandle*

**mock\_chroots**

**Return type** *MockChrootHandle*

**classmethod create\_from\_params** (*root\_url=None, login=None, token=None*)

Create client instance using the given parameters

**Parameters**

- **root\_url** (*str*) – Url to the Copr service, default: “http://copr.fedoraproject.org”
- **login** (*str*) – api login
- **token** (*str*) – api token

**Return type** *client\_v2.client.CoprClient*

**classmethod create\_from\_file\_config** (*filepath=None, ignore\_error=False*)

Creates Copr client using the information from the config file.

**Parameters**

- **filepath** (*str*) – specifies config location, default: “~/config/copr”
- **ignore\_error** (*bool*) – When true and config is missing, creates default Client without credentials

**Return type** *client\_v2.client.CoprClient*

**post\_init** ()

Finalizes client initialization by querying API root info

## Handlers

### Project

**class** `copr.client_v2.handlers.ProjectHandle` (*client, nc, root\_url, projects\_href*)

**get\_list** (*search\_query=None, owner=None, name=None, limit=None, offset=None*)

Retrieves projects object according to the given parameters

**Parameters**

- **search\_query** (*str*) – search projects with such string
- **owner** (*str*) – owner username
- **name** (*str*) – project name
- **limit** (*int*) – limit number of projects
- **offset** (*int*) – number of projects to skip

**Return type** *ProjectList*

**get\_one** (*project\_id*)

Retrieves project object.

**Parameters** `project_id` (*int*) – project identifier

**Return type** *Project*

**create** (*name, owner, chroots, description=None, instructions=None, homepage=None, contact=None, disable\_createrepo=None, build\_enable\_net=None, repos=None*)  
Creates new project

**Parameters**

- **name** – project name
- **owner** – username
- **chroots** – list of mock chroot to be used in project
- **description** –
- **instructions** –
- **homepage** –
- **contact** –
- **disable\_createrepo** (*bool*) –
- **build\_enable\_net** (*bool*) –
- **repos** – list of additional repos enabled for builds

**Return type** *Project*

**update** (*project\_entity*)  
Updates project.

**Parameters** `project_entity` (*ProjectEntity*) – project entity to use for update

**Return type** *OperationResult*

**delete** (*project\_id*)  
Deletes project.

**Parameters** `project_id` (*int*) – project identifier

**Return type** *OperationResult*

**get\_builds\_handle** ()

**Return type** *BuildHandle*

**get\_build\_tasks\_handle** ()

**Return type** *BuildTasksHandle*

**get\_project\_chroots\_handle** ()

**Return type** *ProjectChrootHandle*

## Project chroot

**class** `copr.client_v2.handlers.ProjectChrootHandle` (*client, nc, root\_url*)

**get\_base\_url** (*project, \*\*kwargs*)

**get\_one** (*project, name*)

Retrieves project chroot object.

**Parameters**

- **project** (*Project*) – parent project for the chroot
- **name** (*str*) – chroot name

**Return type** *ProjectChroot*

**get\_list** (*project*)

Retrieves project chroot list object.

**Parameters** **project** (*Project*) – parent project for the chroot

**Return type** *ProjectChrootList*

**disable** (*project, name*)

Disables one chroot for the project

**Parameters**

- **project** (*Project*) – parent project for the chroot
- **name** (*str*) – chroot name to disable

**enable** (*project, name, buildroot\_pkgs=None*)

Enables one chroot for the project

**Parameters**

- **project** (*Project*) – parent project for the chroot
- **name** (*str*) – chroot name to enable

**Params** **buildroot\_pkgs** packages to add into the buildroot

**Return type** *OperationResult*

**update** (*project, chroot\_entity*)

**Parameters** **chroot\_entity** (*entities.ProjectChrootEntity*) – Entity to update

**Return type** *OperationResult*

## Build

**class** `copr.client_v2.handlers.BuildHandle` (*client, nc, root\_url, builds\_href*)

**get\_one** (*build\_id*)

Retrieves builds object

**Parameters** **build\_id** (*int*) – id of the target build

**Return type** *Build*

**get\_list** (*project\_id=None, owner=None, limit=None, offset=None*)

Retrieves builds object according to the given parameters

**Parameters**

- **owner** – name of the project owner
- **project\_id** – id of the project

- **limit** – limit number of builds
- **offset** – number of builds to skip

**Return type** *BuildList*

**cancel** (*build\_entity*)

    Cancels the given build

**Parameters** **build\_entity** (*BuildEntity*) – build entity to delete

**Return type** *OperationResult*

**delete** (*build\_id*)

    Deletes the given build

**Parameters** **build\_id** (*int*) – build id to delete

**Return type** *OperationResult*

**create\_from\_url** (*project\_id, srpm\_url, chroots=None, enable\_net=True*)

    Creates new build using public url to the srpm file

**Parameters**

- **project\_id** (*int*) – id of the project where we want to submit new build
- **srpm\_url** (*str*) – url to the source rpm
- **chroots** (*list*) – which chroots should be used during the build
- **enable\_net** (*bool*) – allows to disable network access during the build, default: True

**Returns** created build

**Return type** *Build*

**create\_from\_file** (*project\_id, file\_path=None, file\_obj=None, file\_name=None, chroots=None, enable\_net=True*)

    Creates new build using srpm upload, please specify either *file\_path* or (*file\_obj*, *file\_name*).

**Parameters**

- **project\_id** (*int*) – id of the project where we want to submit new build
- **file\_path** (*str*) – path to the srpm file
- **file\_obj** (*file*) – file-like object to read from
- **file\_name** (*str*) – name for the uploaded file
- **chroots** (*list*) – which chroots should be used during the build
- **enable\_net** (*bool*) – allows to disable network access during the build, default: True

**Returns** created build

**Return type** *Build*

**get\_build\_tasks\_handle** ()

**Return type** *BuildTasksHandle*

## Build task

```
class copr.client_v2.handlers.BuildTaskHandle (client, nc, root_url, build_tasks_href)
```



**get\_list** (*owner=None, project\_id=None, build\_id=None, state=None, offset=None, limit=None*)  
Retrieves build tasks list according to the given parameters

#### Parameters

- **owner** (*str*) – build tasks from the project owner by this user
- **project\_id** (*int*) – get tasks only from this project, when used query parameter `owner` is ignored
- **build\_id** (*int*) – get tasks only from this build, when used query parameters `owner` and `project_id` are ignored
- **state** (*str*) – get build tasks only with this state, allowed values: `failed`, `succeeded`, `canceled`, `running`, `pending`, `starting`, `importing`
- **limit** (*int*) – limit number of projects
- **offset** (*int*) – number of projects to skip

**Return type** *BuildTaskList*

**get\_one** (*build\_id, chroot\_name*)  
Retrieves single build task object

#### Parameters

- **build\_id** (*int*) – id of the build
- **chroot\_name** (*str*) – name of the build chroot

**Return type** *BuildTask*

## Mock chroot

**class** `copr.client_v2.handlers.MockChrootHandle` (*client, nc, root\_url, href*)

**get\_one** (*name*)  
Retrieves mock chroot object.

**Parameters** **name** (*str*) – chroot name

**Return type** *MockChroot*

**get\_list** (*active\_only=True*)  
Retrieves mock chroot list object.

**Parameters** **active\_only** (*bool*) – when `True`, shows only chroots which can be used for builds

**Return type** *MockChrootList*

## Resources

### Project

see entity attributes at *Project entity attributes*.

**class** `copr.client_v2.resources.Project` (*entity, handle, \*\*kwargs*)

**update ()**

Updates project using the current state.

Shortcut for for *ProjectHandle.update ()*

**Return type** *OperationResult*

**delete ()**

Updates project using the current state

**Return type** *OperationResult*

**get\_self ()**

Retrieves fresh project object from the service

**Return type** *Project*

**get\_builds (\*\*query\_options)**

Get builds owned by this project

**Parameters** **query\_options** – see *handlers.BuildHandle.get\_list ()*

**Return type** *BuildsList*

**get\_build\_tasks (\*\*query\_options)**

Get build tasks owned by this project

**Parameters** **query\_options** – see *handlers.BuildHandle.get\_list ()*

**Return type** *BuildTasksList*

**get\_project\_chroot (name)**

Retrieves project chroot object by the given name

**Parameters** **name (str)** – mock chroot name

**Return type** *ProjectChroot*

**get\_project\_chroot\_list ()**

Retrieves project chroots list

**Return type** *ProjectChrootList*

**enable\_project\_chroot (name)**

Enables given chroot for this project

Shortcut for for *ProjectChrootHandle.enable ()*

**Parameters** **name (str)** – mock chroot name

**Return type** *OperationResult*

**create\_build\_from\_file (\*args, \*\*kwargs)**

Shortcut for *BuildHandle.create\_from\_file ()* (here you don't need to specify *project\_id*)

**create\_build\_from\_url (\*args, \*\*kwargs)**

Shortcut for *BuildHandle.create\_from\_file ()* (here you don't need to specify *project\_id*)

**class** *copr.client\_v2.resources.ProjectList (handle, \*\*kwargs)*

**next\_page ()**

Retrieves next chunk of the Project list for the same query options

**Return type** *ProjectList*

**projects**

**Return type** list of *Project*

## Project chroot

see entity attributes at *Project chroot entity attributes*.

**class** `copr.client_v2.resources.ProjectChroot` (*entity, handle, project, \*\*kwargs*)

**disable** ()

Disables chroot for the bound project

**Return type** *OperationResult*

**update** ()

Updates chroot with the current entity state

**Return type** *OperationResult*

**class** `copr.client_v2.resources.ProjectChrootList` (*handle, project, \*\*kwargs*)

List of the *ProjectChroot* in the one Project.

**chroots**

**Return type** list of *ProjectChroot*

**enable** (*name*)

Enables mock chroot for the current project

**Return type** *OperationResult*

## Build

see entity attributes at *Build entity attributes*.

**class** `copr.client_v2.resources.Build` (*entity, handle, \*\*kwargs*)

**get\_self** ()

Retrieves fresh build object from the service

**Return type** *Build*

**cancel** ()

Updates the current build

**Return type** *OperationResult*

**delete** ()

Deletes the current build

**Return type** *OperationResult*

**get\_build\_tasks** (*\*\*query\_options*)

Get build tasks owned by this build

**Parameters** *query\_options* – see *handlers.BuildHandle.get\_list* ()

**Return type** *BuildTasksList*

**class** `copr.client_v2.resources.BuildList` (*handle*, *\*\*kwargs*)

**next\_page** ()

Retrieves next chunk of the Build list for the same query options

**Return type** *BuildList*

**builds**

**Return type** *BuildList*

## Build task

see entity attributes at *Build task entity attributes*.

**class** `copr.client_v2.resources.BuildTask` (*entity*, *handle*, *\*\*kwargs*)

**get\_self** ()

Retrieves fresh build task object from the service

**Return type** *Build*

**class** `copr.client_v2.resources.BuildTaskList` (*handle*, *\*\*kwargs*)

List of build tasks

**build\_tasks**

**Return type** list of *BuildTask*

## Mock chroot

see entity attributes at *Mock chroot entity attributes*.

**class** `copr.client_v2.resources.MockChroot` (*entity*, *handle*, *\*\*kwargs*)

**class** `copr.client_v2.resources.MockChrootList` (*handle*, *\*\*kwargs*)

List of the mock chroots supported by the service

**chroots**

**Return type** list of *MockChroot*

## Operation result

**class** `copr.client_v2.resources.OperationResult` (*handle*, *response=None*, *entity=None*,  
*options=None*, *expected\_status=200*)

Fake resource to represent results of the requested operation

**new\_location**

Contains an url to the new location produced by an operation If operation doesn't produce a new location would contain None

**Return type** str

**is\_successful()**

Performs check if status code is equal to the expected value of particular request.

**Return type** bool

## Entities

Represents domain objects like project or build. The user of `python-copr` is not expected to use the directly.

**class** `copr.client_v2.entities.Link` (*role, href*)

Bases: `copr.util.UnicodeMixin`

**classmethod** `from_dict` (*data\_dict*)

**class** `copr.client_v2.entities.Entity` (*\*\*kwargs*)

Bases: `copr.util.UnicodeMixin`

**to\_dict** ()

**to\_json** ()

**classmethod** `from_dict` (*raw\_dict*)

**class** `copr.client_v2.entities.ProjectEntity` (*\*\*kwargs*)

Bases: `copr.client_v2.entities.Entity`

**class** `copr.client_v2.entities.ProjectCreateEntity` (*\*\*kwargs*)

Bases: `copr.client_v2.entities.Entity`

**class** `copr.client_v2.entities.ProjectChrootEntity` (*\*\*kwargs*)

Bases: `copr.client_v2.entities.Entity`

**class** `copr.client_v2.entities.BuildEntity` (*\*\*kwargs*)

Bases: `copr.client_v2.entities.Entity`

**is\_finished** ()

Check is the build was finished

**Return type** bool

**class** `copr.client_v2.entities.BuildTaskEntity` (*\*\*kwargs*)

Bases: `copr.client_v2.entities.Entity`

**class** `copr.client_v2.entities.MockChrootEntity` (*\*\*kwargs*)

Bases: `copr.client_v2.entities.Entity`

## 3.3 Client version 3

This documentation describes the third version of python Copr client, which officially supports APIv3. This version addresses the problems that we had with previous versions and therefore obsoletes them.

Operations are separated among multiple proxy classes. All methods are just tiny wrappers around API endpoints. When a method is successful, it returns Munch with data as a result. Otherwise, an exception is raised.

### 3.3.1 Example usage

```
from copr.v3 import Client

# Create an API client from config file
client = Client.create_from_config_file()

# Create a new project
chroots = ["fedora-rawhide-x86_64", "fedora-rawhide-i386"]
client.project_proxy.add("@copr", "foo", chroots, description="Some desc")

# Build a package
url = "http://foo.ex/bar.src.rpm"
build = client.build_proxy.create_from_url("@copr", "foo", url)
print(build.id)
```

### 3.3.2 Quick overview

#### Client initialization

Before an API client can be used, it needs to be initialized with a configuration. There are several ways to do it. The most standard option is reading the `~/.config/copr` file and providing it to the `Client` class. Please read <https://copr.fedorainfracloud.org/api/> for more information about API token.

```
from copr.v3 import Client
client = Client.create_from_config_file()
pprint(client.config)
```

```
{'copr_url': u'https://copr.fedorainfracloud.org',
 'login': u'secretlogin',
 'token': u'secrettoken',
 'username': u'frostyx'}
```

A different config file can be easily used by passing its path to `create_from_config_file` method.

```
client = Client.create_from_config_file("/some/alternative/copr")
```

It is not required to use a configuration stored in a file though. Configuration dict can be passed to the `Client` constructor.

```
config = {'copr_url': u'https://copr.fedorainfracloud.org',
          'login': u'secretlogin',
          'token': u'secrettoken',
          'username': u'frostyx'}

client = Client(config)
assert client.config == config
```

Similarly it can be done when using proxies directly.

```
from copr.v3 import BuildProxy
build_proxy = BuildProxy.create_from_config_file()
```

Or even without configuration file.

```
config = {'copr_url': u'https://copr.fedorainfracloud.org',
          'login': u'secretlogin',
          'token': u'secrettoken',
          'username': u'frostyx'}

build_proxy = BuildProxy(config)
```

And finally, it is possible to just read the configuration file.

```
from copr.v3 import config_from_file
config = config_from_file()
client = Client(config)
```

## Data structures

A data returned from successful API calls are transformed and presented to you as a Munch (it is a subclass of a dict with all its features, with an additional support of accessing its attributes like object properties, etc). This page shows how to work with the results, how to access the original responses from frontend and what are the specifics for lists of results.

First, let's just initialize an API client, and obtain some object (in this example a build) to be examined.

```
from copr.v3 import Client
client = Client.create_from_config_file()
build = client.build_proxy.get(2545)
pprint(build)
```

As advertised, the data is represented as a Munch.

```
Munch({'source_package': {'url': u'http://backend/results/@copr/copr/srpm-builds/
↳00002545/ed-1.14.2-3.fc26.src.rpm', u'version': u'1.14.2-3.fc26', u'name': u'ed'},
↳'__response__': <Response [200]>, u'projectname': u'copr', u'started_on':
↳1526406595, u'submitted_on': 1525635534, u'state': u'succeeded', u'ended_on':
↳1526408106, u'ownername': u'@copr', u'repo_url': u'http://backend/results/@copr/copr
↳', u'submitter': u'frostyx', u'chroots': [u'fedora-27-x86_64', u'fedora-rawhide-x86_
↳64'], u'id': 2545})
```

What exactly it is? It is a structure that extends dict and add more features to it.

```
print(type(build))
print(isinstance(build, dict))
```

```
<class 'munch.Munch'>
True
```

In the first example, it is quite hard to see, what attributes are available and what are their values. It is possible to print the structure in more human-readable format.

```
from pprint import pprint
pprint(dict(build))
```

```
{'__response__': <Response [200]>,
 u'chroots': [u'fedora-27-x86_64', u'fedora-rawhide-x86_64'],
 u'ended_on': 1526408106,
 u'id': 2545,
```

(continues on next page)

(continued from previous page)

```

u'ownername': u'@copr',
u'projectname': u'copr',
u'repo_url': u'http://backend/results/@copr/foo',
u'source_package': {u'name': u'ed',
                    u'url': u'http://backend/results//@copr/copr/srpm-builds/
↪00002545/ed-1.14.2-3.fc26.src.rpm',
                    u'version': u'1.14.2-3.fc26'}},
u'started_on': 1526406595,
u'state': u'succeeded',
u'submitted_on': 1525635534,
u'submitter': u'frostyx'}

```

Attributes are accessible through standard dict bracket-style, but also through object like property-style.

```

assert build.ownername == build["ownername"]
print (build.ownername)

```

```
@copr
```

Every data munch also stores the original response from frontend.

```

print (build.__response__)
print (type(build.__response__))
print (build.__response__.status_code)

```

```

<Response [200]>
requests.models.Response
200

```

## Lists of objects

Now, it should be clear how a single data object is represented. Let's see how the situation looks like when multiple objects are returned.

```

builds = client.build_proxy.get_list("@copr", "copr")
print (builds)

```

At the first sight, it is just a list of munches.

```

[Munch({u'source_package': {u'url': u'http://backend/results//@copr/copr/srpm-builds/
↪00002544/mksh-56c-3.fc26.src.rpm', u'version': u'56c-3.fc26', u'name': u'mksh'}, u
↪'projectname': u'copr', u'started_on': 1519063348, u'submitted_on': 1519062565, u
↪'state': u'succeeded', u'ended_on': 1519064069, u'ownername': u'frostyx', u'repo_url
↪': u'http://backend/results/@copr/copr', u'submitter': u'frostyx', u'chroots': [u
↪'fedora-rawhide-i386', u'fedora-rawhide-x86_64'], u'id': 2544}),
Munch({u'source_package': {u'url': u'http://backend/results//@copr/copr/srpm-builds/
↪00002545/ed-1.14.2-3.fc26.src.rpm', u'version': u'1.14.2-3.fc26', u'name': u'ed'}, u
↪'projectname': u'copr', u'started_on': 1526406595, u'submitted_on': 1525635534, u
↪'state': u'succeeded', u'ended_on': 1526408106, u'ownername': u'@copr', u'repo_url
↪': u'http://backend/results/@copr/copr', u'submitter': u'frostyx', u'chroots': [u
↪'fedora-27-x86_64', u'fedora-rawhide-x86_64'], u'id': 2545})]

```

Not exactly. It is a subclass of a list created in the copr package.



```
print(type(builds))
print(isinstance(builds, list))
```

```
<class 'copr.v3.helpers.List'>
True
```

Let's answer the anticipated question, why do we need a modified implementation of a list. It can provide the frontend response in a `__response__` attribute the same way that single munch does.

```
print(builds.__response__)
<Response [200]>
```

It also provides a meta attribute, that has information about ordering results in the list and possibly limiting their number. Please read more about the pagination.

```
print(builds.meta)
Munch({'u'offset': 0, 'u'limit': None, 'u'order_type': u'ASC', 'u'order': u'id'})
```

Iterating through all objects in the response looks as expected.

```
for build in builds:
    print("Build {} {}".format(build.id, build.state))
```

```
Build 2544 succeeded
Build 2545 succeeded
```

## Modifying data

Previous examples show the data structures when the object was explicitly queried (i.e. `get` or `get_list` method was used). It remains to be explained, how the responses look like when a user tries to add, modify, or delete some object. Simply enough, the operation is executed and the object is implicitly queried afterward.

```
build = client.build_proxy.delete(2545)
print(build)
```

```
Munch({'u'source_package': {'u'url': u'http://backend/results/@copr/copr/srpm-builds/
↳ 00002545/ed-1.14.2-3.fc26.src.rpm', 'u'version': u'1.14.2-3.fc26', 'u'name': u'ed'}, u
↳ 'projectname': u'copr', 'u'started_on': 1526406595, 'u'submitted_on': 1525635534, u
↳ 'state': u'succeeded', 'u'ended_on': 1526408106, 'u'ownername': u'@copr', 'u'repo_url
↳ ': u'http://backend/results/@copr/copr', 'u'submitter': u'frostyx', 'u'chroots': [u
↳ 'fedora-27-x86_64', 'u'fedora-rawhide-x86_64'], 'u'id': 2545})
```

The object was deleted, so it obviously can't be queried one more time

```
client.build_proxy.get(build.id)
```

```
CoprNoResultException: Build 2545 does not exist.
```

## Error handling

All methods from proxy classes return Munch with data only when the API call succeeds. Otherwise, an exception is raised.

This example code tries to cancel a build. Such thing is possible only when the build is not already finished.

```
from copr.v3 import Client
client = Client.create_from_config_file()

try:
    build = client.build_proxy.cancel(123)
    print("Build {} is {}".format(build.id, build.state))
except CoprRequestException as ex:
    print(ex)
```

In case that the build can be canceled, we get this output.

```
Build 123 is canceled
```

Otherwise, an exception is raised and handled.

```
Cannot cancel build 123
```

### Debugging

Sometimes it is useful to dig deeper and examine the failure. Exceptions contain a `result` attribute returning a Munch with additional information.

```
except CoprRequestException as ex:
    print(ex)
    print(ex.result)
```

```
Cannot cancel build 123
Munch({'__response__': <Response [500]>, 'error': u'Cannot cancel build 123'})
```

The stored response is a standard `requests.Response` so every possible detail like status code or headers can be examined.

```
except CoprRequestException as ex:
    print(type(ex.result.__response__))
    print(ex.result.__response__)
    print(ex.result.__response__.status_code)
    print(ex.result.__response__.headers)
```

```
<class 'requests.models.Response'>
<Response [500]>
500
{'Date': 'Wed, 25 Jul 2018 21:40:48 GMT', 'Content-Length': '42', 'Content-Type':
↪ 'application/json', 'Server': 'Werkzeug/0.12.2 Python/3.6.4'}
```

### Status codes

An appropriate status codes are used for specific situations.

| Status code | Reason   |
|-------------|--|
| 200         | Successful request   |
| 401         | Unauthorized request when login is required                        |
| 403         | Insufficient permissions, e.g. modifying a project of someone else |
| 404         | API endpoint or requested object was not found                     |
| 500         | General frontend error   |

## Exception hierarchy

## Pagination

Pagination is not enforced, so users don't need to care about it if they don't want to. In this code sample, all packages from given project are returned, regardless their number.

```
from copr.v3 import Client
client = Client(config)

packages = client.package_proxy.get_list("@copr", "copr")
print(packages)
```

```
[Munch({'id': 1, 'ownername': '@copr', 'projectname': 'copr', 'state': 'pending', ...}
↪),
Munch({'id': 2, 'ownername': '@copr', 'projectname': 'copr', 'state': 'pending', ...}
↪),
Munch({'id': 3, 'ownername': '@copr', 'projectname': 'copr', 'state': 'importing', ..
↪.}),
Munch({'id': 4, 'ownername': '@copr', 'projectname': 'copr', 'state': 'importing', ..
↪.}),
Munch({'id': 5, 'ownername': '@copr', 'projectname': 'copr', 'state': 'canceled', ...
↪})]
```

However, in some cases, it may be useful to obtain just a limited number of objects. Querying all builds from a project with hundreds of thousands of them may be painfully slow or even timeout. And if all of them are not even needed, it is a huge waste of resources. It all depends on the specific use-case. When it is useful, you can query limited and/or ordered number of objects.

```
pagination = {"limit": 3, "order": "name"}
packages = client.package_proxy.get_list("@copr", "copr", pagination=pagination)
print(packages)
print(packages.meta)
```

```
[Munch({'id': 1, 'ownername': '@copr', 'projectname': 'copr', 'state': 'pending', ...}
↪),
Munch({'id': 2, 'ownername': '@copr', 'projectname': 'copr', 'state': 'pending', ...}
↪),
Munch({'id': 3, 'ownername': '@copr', 'projectname': 'copr', 'state': 'importing', ..
↪.})]
Munch({'u'offset': 0, u'limit': 3, u'order_type': u'ASC', u'order': u'id'})
```

And finally, in some cases, it may be useful to iterate through all objects, but not obtaining them all at once (e.g. when projects are so large, that requesting everything timeouts).

```
from copr.v3 import next_page

package_page = client.package_proxy.get_list("@copr", "copr", pagination={"limit": 3})
while package_page:
    for package in package_page:
        print(package)
    print("---")
    package_page = next_page(package_page)
```

```
Munch({'id': 1, 'ownername': '@copr', 'projectname': 'copr', 'state': 'pending', ...})
Munch({'id': 2, 'ownername': '@copr', 'projectname': 'copr', 'state': 'pending', ...})
Munch({'id': 3, 'ownername': '@copr', 'projectname': 'copr', 'state': 'importing', ...}
↪)
---
Munch({'id': 4, 'ownername': '@copr', 'projectname': 'copr', 'state': 'importing', ...}
↪)
Munch({'id': 5, 'ownername': '@copr', 'projectname': 'copr', 'state': 'canceled', ...}
↪)
```

### Pagination parameters

| Field      | Type | Description                              |
|------------|------|--|
| limit      | int  | number of objects to obtain              |
| offset     | int  | number of objects from beginning to skip |
| order      | str  | sort objects by this property            |
| order_type | str  | “ASC” or “DESC”                          |

### Working with proxies directly

In all sample codes, it is used Client to provide proxy objects (`project_proxy`, `build_proxy`, etc). However, that is not the only way how to do it. Proxy classes can be also initialized directly.

Following code samples are equal.

```
from copr.v3 import Client
client = Client.create_from_config_file()
build = client.build_proxy.get(123)
```

Same thing without using Client.

```
from copr.v3 import BuildProxy
config = {"username": "frostyx", , "copr_url": "https://copr.fedorainfracloud.org/",
         "login": "somehash", "token": "someotherhash"}
build_proxy = BuildProxy(config)
build = build_proxy.get(123)
```

### 3.3.3 Resources info

## Proxies

User interface is separated among multiple proxy classes such as `ProjectProxy`, `BuildProxy`, etc, which provide API operations on given resource.

There are several methods available across majority (there are some corner cases where it wouldn't make sense) of proxies. Naturally, all methods (e.g. `auth_check`) from `BaseProxy` are available everywhere. Moreover, proxies implement `get` method to get one specific object and `get_list` to get multiple objects that meet some criteria (e.g. all successful builds from a project). When it makes sense, proxies also implement an `edit` method that modifies an object. Exception for this is for example, a `BuildProxy` because it shouldn't be possible to change a build. Similarly, most of the proxies have a `delete` method except for e.g. `BuildChrootProxy`.

## Base

## Project

## Build

## Package

## Module

## Project Chroot

## Build Chroot

## 3.3.4 Parameters

### Build Options

When submitting a new build, it is possible to specify some options in `buildopts` parameter. Those are common for all build methods.

| Field                                  | Type              | Description  |
|--|-------------------|--|
| <code>timeout</code>                   | <code>int</code>  | build timeout  |
| <code>memory</code>                    | <code>int</code>  | amount of required memory for build process  |
| <code>chroots</code>                   | list of strings   | build only for given chroots   |
| <code>background</code>                | <code>bool</code> | mark the build as a background job   |
| <code>progress_callbackcallable</code> |                   | function that receives a <code>MultipartEncoderMonitor</code> instance for each chunk of uploaded data |

### Example usage

```
url = "http://foo.ex/baz.src.rpm"
client.build_proxy.create_from_url(url, buildopts={
    "chroots": ["fedora-rawhide-x86_64"],
    "background": True,
})
```

## Package Source Type

Read more about source types in the [User Documentation](#).

### SCM

Parameters when `source_type=scm` is used. See [User Documentation](#) for more information.

| Field                            | Type | Description  |
|----------------------------------|------|--|
| <code>clone_url</code>           | str  |  |
| <code>committish</code>          | str  |  |
| <code>subdirectory</code>        | str  |  |
| <code>spec</code>                | str  |  |
| <code>scm_type</code>            | str  | “git”, “svn”   |
| <code>source_build_method</code> | str  | See <a href="#">User documentation</a> for more info |

### Rubygems

Parameters when `source_type=rubygems` is used. See [User Documentation](#) for more information.

| Field                 | Type | Description |
|-----------------------|------|-------------|
| <code>gem_name</code> | str  |             |

### PyPI

Parameters when `source_type=pypi` is used. See [User Documentation](#) for more information.

| Field                             | Type        | Description |
|-----------------------------------|-------------|-------------|
| <code>pypi_package_name</code>    | str         |             |
| <code>pypi_package_version</code> | str         |             |
| <code>spec_template</code>        | str         |             |
| <code>python_versions</code>      | list of int |             |

### Custom

Parameters when `source_type=custom` is used. See [User Documentation](#) for more information.

| Field                  | Type | Description |
|------------------------|------|-------------|
| <code>script</code>    | str  |             |
| <code>builddeps</code> | str  |             |
| <code>resultdir</code> | str  |             |
| <code>chroot</code>    | str  |             |

### Example Usage

```
client.package_proxy.add("@copr", "foo", "mypackage",
                          source_type="rubygems",
                          source={"gem_name": "mygem"})
```

Both Legacy client and Client version 2 are now obsolete, please migrate to Client version 3.





## CHAPTER 4

---

### Indices and tables

---

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



**C**

`copr.client_v2.entities`, 33



**B**

BaseHandle (class in *copr.client.responses*), 13  
 Build (class in *copr.client\_v2.resources*), 31  
 build\_tasks (*copr.client\_v2.client.CoprClient* attribute), 25  
 build\_tasks (*copr.client\_v2.resources.BuildTaskList* attribute), 32  
 BuildEntity (class in *copr.client\_v2.entities*), 33  
 BuildHandle (class in *copr.client.responses*), 14  
 BuildHandle (class in *copr.client\_v2.handlers*), 27  
 BuildList (class in *copr.client\_v2.resources*), 31  
 builds (*copr.client\_v2.client.CoprClient* attribute), 24  
 builds (*copr.client\_v2.resources.BuildList* attribute), 32  
 BuildTask (class in *copr.client\_v2.resources*), 32  
 BuildTaskEntity (class in *copr.client\_v2.entities*), 33  
 BuildTaskHandle (class in *copr.client\_v2.handlers*), 28  
 BuildTaskList (class in *copr.client\_v2.resources*), 32  
 BuildWrapper (class in *copr.client.responses*), 14

**C**

cancel() (*copr.client\_v2.handlers.BuildHandle* method), 28  
 cancel() (*copr.client\_v2.resources.Build* method), 31  
 cancel\_build() (*copr.client.client.CoprClient* method), 10  
 cancel\_build() (*copr.client.responses.BuildHandle* method), 14  
 chroots (*copr.client\_v2.resources.MockChrootList* attribute), 32  
 chroots (*copr.client\_v2.resources.ProjectChrootList* attribute), 31  
 copr.client\_v2.entities (module), 33  
 CoprClient (class in *copr.client.client*), 9  
 CoprClient (class in *copr.client\_v2.client*), 24  
 CoprResponse (class in *copr.client.responses*), 13  
 create() (*copr.client\_v2.handlers.ProjectHandle*

method), 26  
 create\_build\_from\_file() (*copr.client\_v2.resources.Project* method), 30  
 create\_build\_from\_url() (*copr.client\_v2.resources.Project* method), 30  
 create\_client2\_from\_file\_config() (in module *copr*), 24  
 create\_client2\_from\_params() (in module *copr*), 24  
 create\_from\_file() (*copr.client\_v2.handlers.BuildHandle* method), 28  
 create\_from\_file\_config() (*copr.client.client.CoprClient* static method), 9  
 create\_from\_file\_config() (*copr.client\_v2.client.CoprClient* class method), 25  
 create\_from\_params() (*copr.client\_v2.client.CoprClient* class method), 25  
 create\_from\_url() (*copr.client\_v2.handlers.BuildHandle* method), 28  
 create\_new\_build() (*copr.client.client.CoprClient* method), 10  
 create\_new\_build() (*copr.client.responses.ProjectHandle* method), 14  
 create\_project() (*copr.client.client.CoprClient* method), 11

**D**

delete() (*copr.client\_v2.handlers.BuildHandle* method), 28  
 delete() (*copr.client\_v2.handlers.ProjectHandle* method), 26  
 delete() (*copr.client\_v2.resources.Build* method), 31

`delete()` (*copr.client\_v2.resources.Project method*), 30  
`delete_project()` (*copr.client.client.CoprClient method*), 11  
`delete_project()` (*copr.client.responses.ProjectHandle method*), 14  
`disable()` (*copr.client\_v2.handlers.ProjectChrootHandle method*), 27  
`disable()` (*copr.client\_v2.resources.ProjectChroot method*), 31

**E**

`enable()` (*copr.client\_v2.handlers.ProjectChrootHandle method*), 27  
`enable()` (*copr.client\_v2.resources.ProjectChrootList method*), 31  
`enable_project_chroot()` (*copr.client\_v2.resources.Project method*), 30  
 Entity (class in *copr.client\_v2.entities*), 33

**F**

`from_dict()` (*copr.client\_v2.entities.Entity class method*), 33  
`from_dict()` (*copr.client\_v2.entities.Link class method*), 33

**G**

`get_base_url()` (*copr.client\_v2.handlers.ProjectChrootHandle method*), 26  
`get_build_details()` (*copr.client.client.CoprClient method*), 10  
`get_build_details()` (*copr.client.responses.BuildHandle method*), 14  
`get_build_tasks()` (*copr.client\_v2.resources.Build method*), 31  
`get_build_tasks()` (*copr.client\_v2.resources.Project method*), 30  
`get_build_tasks_handle()` (*copr.client\_v2.handlers.BuildHandle method*), 28  
`get_build_tasks_handle()` (*copr.client\_v2.handlers.ProjectHandle method*), 26  
`get_builds()` (*copr.client\_v2.resources.Project method*), 30  
`get_builds_handle()` (*copr.client\_v2.handlers.ProjectHandle method*), 26  
`get_list()` (*copr.client\_v2.handlers.BuildHandle method*), 27  
`get_list()` (*copr.client\_v2.handlers.BuildTaskHandle method*), 28  
`get_list()` (*copr.client\_v2.handlers.MockChrootHandle method*), 29  
`get_list()` (*copr.client\_v2.handlers.ProjectChrootHandle method*), 27  
`get_list()` (*copr.client\_v2.handlers.ProjectHandle method*), 25  
`get_one()` (*copr.client\_v2.handlers.BuildHandle method*), 27  
`get_one()` (*copr.client\_v2.handlers.BuildTaskHandle method*), 29  
`get_one()` (*copr.client\_v2.handlers.MockChrootHandle method*), 29  
`get_one()` (*copr.client\_v2.handlers.ProjectChrootHandle method*), 26  
`get_one()` (*copr.client\_v2.handlers.ProjectHandle method*), 25  
`get_project_chroot()` (*copr.client\_v2.resources.Project method*), 30  
`get_project_chroot_details()` (*copr.client.client.CoprClient method*), 12  
`get_project_chroot_details()` (*copr.client.responses.ProjectChrootHandle method*), 14  
`get_project_chroot_list()` (*copr.client\_v2.resources.Project method*), 30  
`get_project_chroots_handle()` (*copr.client\_v2.handlers.ProjectHandle method*), 26  
`get_project_details()` (*copr.client.client.CoprClient method*), 11  
`get_project_details()` (*copr.client.responses.ProjectHandle method*), 13  
`get_projects_list()` (*copr.client.client.CoprClient method*), 12  
`get_self()` (*copr.client\_v2.resources.Build method*), 31  
`get_self()` (*copr.client\_v2.resources.BuildTask method*), 32  
`get_self()` (*copr.client\_v2.resources.Project method*), 30

**I**

`is_finished()` (*copr.client\_v2.entities.BuildEntity method*), 33  
`is_successful()` (*copr.client\_v2.resources.OperationResult method*), 33

**L**

Link (class in *copr.client\_v2.entities*), 33

## M

mock\_chroots (*copr.client\_v2.client.CoprClient* attribute), 25

MockChroot (*class in copr.client\_v2.resources*), 32

MockChrootEntity (*class in copr.client\_v2.entities*), 33

MockChrootHandle (*class in copr.client\_v2.handlers*), 29

MockChrootList (*class in copr.client\_v2.resources*), 32

modify\_project() (*copr.client.client.CoprClient* method), 12

modify\_project() (*copr.client.responses.ProjectHandle* method), 13

modify\_project\_chroot\_details() (*copr.client.client.CoprClient* method), 13

modify\_project\_chroot\_details() (*copr.client.responses.ProjectChrootHandle* method), 14

## N

new\_location (*copr.client\_v2.resources.OperationResult* attribute), 32

next\_page() (*copr.client\_v2.resources.BuildList* method), 32

next\_page() (*copr.client\_v2.resources.ProjectList* method), 30

## O

OperationResult (*class in copr.client\_v2.resources*), 32

## P

post\_init() (*copr.client\_v2.client.CoprClient* method), 25

Project (*class in copr.client\_v2.resources*), 29

project\_chroots (*copr.client\_v2.client.CoprClient* attribute), 24

project\_handle (*copr.client.responses.BuildHandle* attribute), 14

ProjectChroot (*class in copr.client\_v2.resources*), 31

ProjectChrootEntity (*class in copr.client\_v2.entities*), 33

ProjectChrootHandle (*class in copr.client.responses*), 14

ProjectChrootHandle (*class in copr.client\_v2.handlers*), 26

ProjectChrootList (*class in copr.client\_v2.resources*), 31

ProjectChrootWrapper (*class in copr.client.responses*), 14

ProjectCreateEntity (*class in copr.client\_v2.entities*), 33

ProjectEntity (*class in copr.client\_v2.entities*), 33

ProjectHandle (*class in copr.client.responses*), 13

ProjectHandle (*class in copr.client\_v2.handlers*), 25

ProjectList (*class in copr.client\_v2.resources*), 30

projects (*copr.client\_v2.client.CoprClient* attribute), 24

projects (*copr.client\_v2.resources.ProjectList* attribute), 30

ProjectWrapper (*class in copr.client.responses*), 14

## S

search\_projects() (*copr.client.client.CoprClient* method), 13

## T

to\_dict() (*copr.client\_v2.entities.Entity* method), 33

to\_json() (*copr.client\_v2.entities.Entity* method), 33

## U

update() (*copr.client\_v2.handlers.ProjectChrootHandle* method), 27

update() (*copr.client\_v2.handlers.ProjectHandle* method), 26

update() (*copr.client\_v2.resources.Project* method), 29

update() (*copr.client\_v2.resources.ProjectChroot* method), 31