PyUploadcare Documentation

Release 1.3.6

Uploadcare LLC

The most important thing for us at Uploadcare is to make file uploading on the web really easy. Everyone is used to the routine work, related to allowing users upload their userpics or attach resumes: from installing image processing libraries to creating folder with right permissions to ensuring the server never goes down or out of space to enabling CDN. Feature like ability to simply use a picture from Facebook or manual cropping are much more burdensome, hence rare. Uploadcare's goal is to change the status quo.

This library consists of an API interface for Uploadcare and a couple of Django goodies.

A simple Uploadcare `FileField` can be added to an existing Django project in just a couple of *simple steps*. As a result, your users are going to be able to see the progress of the upload, choose files from Google Drive or Instagram, and edit form while files are uploading asynchornously.

Contents:

Installation

This part of the documentation covers the installation of PyUploadcare.

1.1 Pip

Installing pyuploadcare is simple with pip:

```
$ pip install pyuploadcare
```

or, if you're into vintage:

```
$ easy_install pyuploadcare
```

1.2 Get the Code

PyUploadcare is developed on GitHub. You can clone the public repository:

```
$ git clone git://github.com/uploadcare/pyuploadcare.git
```

After that you can install it:

```
$ python setup.py install
```

Quickstart

This page gives a good introduction in how to get started with PyUploadcare. This assumes you have already installed PyUploadcare. If you do not, head over to the *Installation* section.

> **Warning:** Keep in mind that Uploadcare signature authentication will fail if computer clock is not synchronized.

2.1 Get API Keys

First of all, you'll need API keys: public and private. You can get them at the Uploadcare website. If you don't have an account yet, you can use demo keys, as in example. However, the files on demo account are regularly deleted, so create an account as soon as Uploadcare catches your fancy.

2.2 How to use it with Django?

Assume you have a Django project with *gallery* app.

2.2.1 Application Setup

Add `pyuploadcare.dj` into `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    # ...
    'pyuploadcare.dj',

    'gallery',
)
```

As soon as you *got your API keys*, add them to your Django settings file:

```
UPLOADCARE = {
    'pub_key': 'demopublickey',
    'secret': 'demoprivatekey',
}
```

Uploadcare image field adding to your `gallery/models.py` is really simple. Like that:

```python
from django.db import models

from pyuploadcare.dj import ImageField


class Photo(models.Model):

    title = models.CharField(max_length=255)
    photo = ImageField()
```

`ImageField` doesn't require any arguments, file paths or whatever. **It just works.** That's the point of it all. It looks nice in the admin interface as well:

Obviously, you would want to use Uploadcare field outside an admin. It's going to work just as well, but, however, you have to remember to add `{{ form.media }}` in the `<head>` tag of your page:

```html
{{ form.media }}

<form action="" method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="Save"/>
</form>
```

This is a default Django form property which is going to render any scripts needed for the form to work, in our case – Uploadcare scripts.

2.2.2 Using it in templates

You can construct url with all effects manually:

```
{% for photo in photos %}
    {{ photo.title }}
    {{ photo.photo.cdn_url }}-/resize/400x300/-/effect/flip/-/effect/grayscale/
{% endfor %}
```

Refer to CDN docs for more information.

2.3 How to use it in command line?

```
$ ucare -h
```

Django Widget

3.1 Settings

Besides required `pub_key`, `secret` settings there are optional settings, for example, `widget_version` or `widget_variant`:

```
UPLOADCARE = {
    'pub_key': 'demopublickey',
    'secret': 'demoprivatekey',
    'widget_version': '2.5.0',
    'widget_variant': 'min',  // without jQuery
    'cdn_base': 'https://cdn.mycompany.com',
}
```

PyUploadcare takes assets from Uploadcare CDN by default, e.g.:

```
<script src="https://ucarecdn.com/widget/x.y.z/uploadcare/uploadcare.full.min.js"></script>
```

If you don't want to use hosted assets you have to turn off this feature:

```
UPLOADCARE = {
    # ...
    'use_hosted_assets': False,
}
```

In this case local assets will be used.

If you want to provide custom url for assets then you have to specify widget url:

```
UPLOADCARE = {
    # ...
    'use_hosted_assets': False,
    'widget_url': 'http://path.to/your/widget.js',
}
```

Uploadcare widget will use default upload handler url, unless you specify:

```
UPLOADCARE = {
    # ...
    'upload_base_url' = 'http://path.to/your/upload/handler',
}
```

3.2 Model Fields

As you will see, with Uploadcare, adding and working with a file field is just as simple as with a TextField. To attach Uploadcare files to a model, you can use a *FileField* or *ImageField*. These fields play by common Django rules. South migrations are supported.

Note: When you call `your_model_form.is_valid()` or call `photo.full_clean()` directly it invokes `File.store()` method automatically. In other cases you should store objects manually, e.g:

```
photo.photo_2x3 = File('a771f854-c2cb-408a-8c36-71af77811f3b')
photo.save()

photo.photo_2x3.store()
```

3.2.1 FileField

`FileField` does not require an uploaded file to be any certain format.

```python
from django.db import models

from pyuploadcare.dj import FileField


class Candidate(models.Model):

    resume = FileField()
```

3.2.2 ImageField

`ImageField` requires an uploaded file to be an image. An optional parameter `manual_crop` enables, if specified, a manual cropping tool: your user can select a part of an image she wants to use. If its value is an empty string, the user can select any part of an image; you can also use values like `"3:4"` or `"200x300"` to get exact proportions or dimensions of resulting image. Consult widget documentation regarding setting up the manual crop:

```python
from django.db import models

from pyuploadcare.dj import ImageField


class Candidate(models.Model):

    photo = ImageField(blank=True, manual_crop="")
```

3.2.3 Advanced widget options

You can pass any widget options via `FileWidget`'s attrs argument:

```python
from django import forms

from pyuploadcare.dj.forms import FileWidget, ImageField
```

```
# optional. provide advanced widget options: https://uploadcare.com/documentation/widget/#configurat
class CandidateForm(forms.Form):
    photo = ImageField(widget=FileWidget(attrs={
        'data-cdn-base': 'https://cdn.super-candidates.com',
        'data-image-shrink': '1024x1024',
    }))
```

3.2.4 FileGroupField

`FileGroupField` allows you to upload more than one file at a time. It stores uploaded files as a group:

```python
from django.db import models

from pyuploadcare.dj import FileGroupField


class Book(models.Model):

    pages = FileGroupField()
```

3.2.5 ImageGroupField

`ImageGroupField` allows you to upload more than one **image** at a time. It stores uploaded images as a group:

```python
from django.db import models

from pyuploadcare.dj import ImageGroupField


class Gallery(models.Model):

    photos = ImageGroupField()
```

Command Line Tool

```
usage: ucare [-h] [--version] [--pub_key PUB_KEY] [--secret SECRET]
             [--api_base API_BASE] [--upload_base UPLOAD_BASE]
             [--no_check_upload_certificate] [--no_check_api_certificate]
             [--api_version API_VERSION]
             {list,get,store,delete,upload_from_url,upload,create_group,sync}
             ...
```

Options:

--version	show program's version number and exit
--pub_key	API key, if not set is read from uploadcare.ini and ~/.uploadcare config files
--secret	API secret, if not set is read from uploadcare.ini and ~/.uploadcare config files
--api_base	API url, can be read from uploadcare.ini and ~/.uploadcare config files. Default value is https://api.uploadcare.com/
--upload_base	Upload API url, can be read from uploadcare.ini and ~/.uploadcare config files. Default value is https://upload.uploadcare.com/
--no_check_upload_certificate=False	Don't check the uploading API server certificate. Can be read from uploadcare.ini and ~/.uploadcare config files.
--no_check_api_certificate=False	Don't check the REST API server certificate. Can be read from uploadcare.ini and ~/.uploadcare config files.
--api_version	API version, can be read from uploadcare.ini and ~/.uploadcare config files. Default value is 0.4

Sub-commands:

list list all files

```
usage: ucare list [-h] [--since SINCE] [--until UNTIL] [--limit LIMIT]
                  [--request_limit REQUEST_LIMIT] [--stored {True,False,None}]
                  [--removed {True,False,None}]
```

Options:

--since	show files uploaded since
--until	show files uploaded until
--limit=100	files to show
--request_limit=100	files per request

--stored	filter stored files	
	Possible choices: True, False, None	
--removed=False	filter removed files	
	Possible choices: True, False, None	

get get file info

```
usage: ucare get [-h] path
```

Positional arguments:

path	file path

store store file

```
usage: ucare store [-h] [--timeout TIMEOUT] [--wait | --nowait]
                   paths [paths ...]
```

Positional arguments:

paths	file(s) path

Options:

--timeout=5	Set wait seconds until operation completed. Default value is 5 seconds
--wait=True	Wait for operation to be completed
--nowait=True	Do not wait for operation to be completed

delete request delete

```
usage: ucare delete [-h] [--timeout TIMEOUT] [--wait | --nowait]
                    paths [paths ...]
```

Positional arguments:

paths	file(s) path

Options:

--timeout=5	Set wait seconds until operation completed. Default value is 5 seconds
--wait=True	Wait for operation to be completed
--nowait=True	Do not wait for operation to be completed

upload_from_url upload file from url

```
usage: ucare upload_from_url [-h] [--store | --nostore] [--info | --noinfo]
                             [--cdnurl] [--timeout TIMEOUT]
                             [--wait | --nowait]
                             url
```

Positional arguments:

url	file url

Options:

--store=False	Store uploaded file
--nostore=True	Do not store uploaded file

--info=False	Get uploaded file info
--noinfo=True	Do not get uploaded file info
--cdnurl=False	Store file and get CDN url.
--timeout=30	Set wait seconds file uploading from url. Default value is 30 seconds
--wait=True	Wait for upload status
--nowait=True	Do not wait for upload status

upload upload file

```
usage: ucare upload [-h] [--store | --nostore] [--info | --noinfo] [--cdnurl]
                    filename
```

Positional arguments:

filename	filename

Options:

--store=False	Store uploaded file
--nostore=True	Do not store uploaded file
--info=False	Get uploaded file info
--noinfo=True	Do not get uploaded file info
--cdnurl=False	Store file and get CDN url.

create_group create file group

```
usage: ucare create_group [-h] paths [paths ...]
```

Positional arguments:

paths	file paths

sync sync files

```
usage: ucare sync [-h] [--replace] [--uuids UUIDS [UUIDS ...]]
                  [--effects EFFECTS]
                  [path]
```

Positional arguments:

path	Local path. It can contains special patterns like: ${uuid} ${effects} ${filename} ${ext} Default is ${uuid}${ext}

Options:

--replace=False	replace exists files
--uuids	list of file's uuids for sync
--effects	apply effects for synced images.Note that effects will apply to images only.For more information look at: https://uploadcare.com/documentation/cdn/ Example: –effects=resize/200x/-/rotate/90/

Deprecated Bits

This part of the documentation contains things that eventually will be deleted.

`PYUPLOADCARE_USE_HOSTED_ASSETS` django setting. Use `UPLOADCARE['use_hosted_assets']` instead.

`PYUPLOADCARE_WIDGET_URL` django setting. Use `UPLOADCARE['widget_url']` instead.

`PYUPLOADCARE_UPLOAD_BASE_URL` django setting. Use `UPLOADCARE['upload_base_url']` instead.

API Reference

6.1 Core API

You can use pyuploadcare in any Python project. At first you need assign your project keys to conf object. After that you will be able to do direct api calls or use api resources:

```python
>>> import pyuploadcare
>>> pyuploadcare.conf.pub_key = '<your public key>'
>>> pyuploadcare.conf.secret = '<your private key>'
>>> f = pyuploadcare.File('6c5e9526-b0fe-4739-8975-72e8d5ee6342')
>>> f.cdn_url
https://ucarecdn.com/6c5e9526-b0fe-4739-8975-72e8d5ee6342/
```

6.1.1 File API Resource

class pyuploadcare.api_resources.**File**(*cdn_url_or_file_id*)

    File resource for working with user-uploaded files.

    It can take file UUID or group CDN url:

```python
>>> file_ = File('a771f854-c2cb-408a-8c36-71af77811f3b')
>>> file_.cdn_url
https://ucarecdn.com/a771f854-c2cb-408a-8c36-71af77811f3b/
>>> print File('https://ucarecdn.com/a771f854-c2cb-408a-8c36-71af77811f3b/-/effect/flip/')
https://ucarecdn.com/a771f854-c2cb-408a-8c36-71af77811f3b/-/effect/flip/
```

    uuid

        File UUID [1], e.g. a771f854-c2cb-408a-8c36-71af77811f3b.

    default_effects

        String of default effects that is used by File.cdn_url, e.g. effect/flip/-/effect/mirror/.

    class **FileFromUrl**(*token*)

        Contains the logic around an upload from url.

        It expects uploading token, for instance:

```python
>>> ffu = FileFromUrl(token='a6a2db73-2aaf-4124-b2e7-039aec022e18')
>>> ffu.info()
{
    "status': "progress",
```

---

[1] Universally unique identifier according to RFC 4122.

```
        "done": 226038,
        "total": 452076
    }
>>> ffu.update_info()
{
    "status": "success",
    "file_id": "63f652fd-3f40-4b54-996c-f17dc7db5bf1",
    "is_stored": false,
    "done": 452076,
    "uuid": "63f652fd-3f40-4b54-996c-f17dc7db5bf1",
    "original_filename": "olympia.jpg",
    "is_image": true,
    "total": 452076,
    "size": 452076
}
>>> ffu.get_file()
<uploadcare.File 63f652fd-3f40-4b54-996c-f17dc7db5bf1>
```

But it could be failed:

```
>>> ffu.update_info()
{
    "status": "error",
    "error": "some error message"
}
```

get_file()
> Returns `File` instance if upload is completed.

info()
> Returns actual information about uploading as `dict`.
>
> First time it makes API request to get information and keeps it for further using.

update_info()
> Updates and returns information by requesting Uploadcare API.

wait(*timeout=30*, *interval=0.3*, *until_ready=False*)

`File.`**cdn_path**(*effects=None*)

`File.`**cdn_url**
> Returns file's CDN url.
>
> Usage example:

```
>>> file_ = File('a771f854-c2cb-408a-8c36-71af77811f3b')
>>> file_.cdn_url
https://ucarecdn.com/a771f854-c2cb-408a-8c36-71af77811f3b/
```

> You can set default effects:

```
>>> file_.default_effects = 'effect/flip/-/effect/mirror/'
>>> file_.cdn_url
https://ucarecdn.com/a771f854-c2cb-408a-8c36-71af77811f3b/-/effect/flip/-/effect/mirror/
```

classmethod `File.`**construct_from**(*file_info*)
> Constructs `File` instance from file information.
>
> For example you have result of `/files/1921953c-5d94-4e47-ba36-c2e1dd165e1a/` API request:

```
>>> file_info = {
        # ...
        'uuid': '1921953c-5d94-4e47-ba36-c2e1dd165e1a',
        # ...
    }
>>> File.construct_from(file_info)
<uploadcare.File 1921953c-5d94-4e47-ba36-c2e1dd165e1a>
```

File.**copy**(*effects=None*, *target=None*)
 Creates File copy

 If target is None, copy file to Uploadcare storage otherwise copy to target associated with project. Add effects to self.default_effects if any.

File.**datetime_removed**()
 Returns file's remove aware *datetime* in UTC format.

 It might do API request once because it depends on info().

File.**datetime_stored**()
 Returns file's store aware *datetime* in UTC format.

 It might do API request once because it depends on info().

File.**datetime_uploaded**()
 Returns file's upload aware *datetime* in UTC format.

 It might do API request once because it depends on info().

File.**delete**()
 Deletes file by requesting Uploadcare API.

File.**filename**()
 Returns original file name, e.g. "olympia.jpg".

 It might do API request once because it depends on info().

File.**info**()
 Returns all available file information as dict.

 First time it makes API request to get file information and keeps it for further using.

File.**is_image**()
 Returns True if the file is an image.

 It might do API request once because it depends on info().

File.**is_ready**()
 Returns True if the file is fully uploaded on S3.

 It might do API request once because it depends on info().

File.**is_removed**()
 Returns True if file is removed.

 It might do API request once because it depends on info().

File.**is_stored**()
 Returns True if file is stored.

 It might do API request once because it depends on info().

File.**mime_type**()
 Returns the file MIME type, e.g. "image/png".

It might do API request once because it depends on `info()`.

File.**size**()
> Returns the file size in bytes.
>
> It might do API request once because it depends on `info()`.

File.**store**()
> Stores file by requesting Uploadcare API.
>
> Uploaded files do not immediately appear on Uploadcare CDN. Let's consider steps until file appears on CDN:
>
> > • first file is uploaded into https://upload.uploadcare.com/;
> >
> > • after that file is available by API and its `is_public`, `is_ready` are `False`. Now you can store it;
> >
> > • `is_ready` will be `True` when file will be fully uploaded on S3.

File.**update_info**()
> Updates and returns file information by requesting Uploadcare API.

classmethod File.**upload**(*file_obj*)
> Uploads a file and returns `File` instance.

classmethod File.**upload_from_url**(*url*)
> Uploads file from given url and returns `FileFromUrl` instance.

classmethod File.**upload_from_url_sync**(*url*, *timeout=30*, *interval=0.3*, *until_ready=False*)
> Uploads file from given url and returns `File` instance.

File.**uuid**

6.1.2 File Group API Resource

class pyuploadcare.api_resources.**FileGroup**(*cdn_url_or_group_id*)
> File Group resource for working with user-uploaded group of files.
>
> It can take group id or group CDN url:

```
>>> file_group = FileGroup('0513dda0-582f-447d-846f-096e5df9e2bb~2')
```

> You can iterate `file_group` or get `File` instance by key:

```
>>> [file_ for file_ in file_group]
[<uploadcare.File 6c5e9526-b0fe-4739-8975-72e8d5ee6342>, None]
>>> file_group[0]
<uploadcare.File 6c5e9526-b0fe-4739-8975-72e8d5ee6342>
>>> len(file_group)
2
```

> But slicing is not supported because `FileGroup` is immutable:

```
>>> file_group[:]
TypeError: slicing is not supported
```

> If file was deleted then you will get `None`:

```
>>> file_group[1]
None
```

id
> Group id, e.g. `0513dda0-582f-447d-846f-096e5df9e2bb~2`.

cdn_url
> Returns group's CDN url.
>
> Usage example:

```
>>> file_group = FileGroup('0513dda0-582f-447d-846f-096e5df9e2bb~2')
>>> file_group.cdn_url
https://ucarecdn.com/0513dda0-582f-447d-846f-096e5df9e2bb~2/
```

classmethod **construct_from**(*group_info*)
> Constructs `FileGroup` instance from group information.

classmethod **create**(*files*)
> Creates file group and returns `FileGroup` instance.
>
> It expects iterable object that contains `File` instances, e.g.:

```
>>> file_1 = File('6c5e9526-b0fe-4739-8975-72e8d5ee6342')
>>> file_2 = File('a771f854-c2cb-408a-8c36-71af77811f3b')
>>> FileGroup.create((file_1, file_2))
<uploadcare.FileGroup 0513dda0-6666-447d-846f-096e5df9e2bb~2>
```

datetime_created()
> Returns file group's create aware *datetime* in UTC format.

datetime_stored()
> Returns file group's store aware *datetime* in UTC format.

file_cdn_urls
> Returns CDN urls of all files from group without API requesting.
>
> Usage example:

```
>>> file_group = FileGroup('0513dda0-582f-447d-846f-096e5df9e2bb~2')
>>> file_group.file_cdn_urls[0]
'https://ucarecdn.com/0513dda0-582f-447d-846f-096e5df9e2bb~2/nth/0/'
```

info()
> Returns all available group information as `dict`.
>
> First time it makes API request to get group information and keeps it for further using.

is_stored()
> Returns `True` if file is stored.
>
> It might do API request once because it depends on `info()`.

store()
> Stores all group's files by requesting Uploadcare API.
>
> Uploaded files do not immediately appear on Uploadcare CDN.

update_info()
> Updates and returns group information by requesting Uploadcare API.

6.1.3 File List API Resource

class pyuploadcare.api_resources.**FileList**(***kwargs*)
> List of File resources.
>
> This class provides iteration over all uploaded files. You can specify:

- since – a datetime object from which objects will be iterated;

- until – a datetime object to which objects will be iterated;

- limit – a total number of objects to be iterated. If not specified, all available objects are iterated;

- stored – True to include only stored files, False to exclude, None is default, will not exclude anything;

- removed – True to include only removed files, False to exclude, None will not exclude anything. The default is False.

If until is specified, the order of items will be reversed. It is impossible to specify since and until at the same time.

Files can't be stored and removed at the same time, such query will always return an empty set. But files can be not stored and not removed (just uploaded files).

Usage example:

```
>>> for f in FileList(removed=None):
>>>     print f.datetime_uploaded()
```

Count objects:

```
>>> print('Number of stored files is', FileList(stored=True).count())
```

since

until

count

stored

removed

base_url = u'/files/'

constructor(*file_info*)
Constructs File instance from file information.

For example you have result of /files/1921953c-5d94-4e47-ba36-c2e1dd165e1a/ API request:

```
>>> file_info = {
        # ...
        'uuid': '1921953c-5d94-4e47-ba36-c2e1dd165e1a',
        # ...
    }
>>> File.construct_from(file_info)
<uploadcare.File 1921953c-5d94-4e47-ba36-c2e1dd165e1a>
```

filters = [(u'stored', None), (u'removed', False)]

6.1.4 Group List API Resource

class pyuploadcare.api_resources.**GroupList**(***kwargs*)
List of FileGroup resources.

This class provides iteration over all groups for project. You can specify:

- since – a datetime object from which objects will be iterated;

- `until` – a datetime object to which objects will be iterated;

- `limit` – a total number of objects to be iterated. If not specified, all available objects are iterated;

If `until` is specified, the order of items will be reversed. It is impossible to specify `since` and `until` at the same time.

Usage example:

```python
>>> from datetime import datetime, timedelta
>>> for f in GroupList(since=datetime.now() - timedelta(weeks=1)):
>>>     print f.datetime_created()
```

Count objects:

```python
>>> print('Number of groups is', GroupList().count())
```

since

until

count

base_url = u'/groups/'

constructor(*group_info*)
    Constructs `FileGroup` instance from group information.

6.1.5 API Clients

Uploadcare REST client.

It is JSON REST request abstraction layer that is used by the `pyuploadcare.api_resources`.

pyuploadcare.api.**rest_request**(*verb*, *path*, *data=None*, *timeout=<object object>*, *retry_throttled=<object object>*)
    Makes REST API request and returns response as `dict`.

It provides auth headers as well and takes settings from `conf` module.

Make sure that given `path` does not contain leading slash.

Usage example:

```python
>>> rest_request('GET', 'files/?limit=10')
{
    'next': 'https://api.uploadcare.com/files/?limit=10&page=2',
    'total': 1241,
    'page': 1,
    'pages': 125,
    'per_page': 10,
    'previous': None,
    'results': [
        # ...
        {
            # ...
            'uuid': 1921953c-5d94-4e47-ba36-c2e1dd165e1a,
            # ...
        },
        # ...
    ]
}
```

pyuploadcare.api.**uploading_request**(*verb*, *path*, *data=None*, *files=None*, *timeout=<object object>*)

    Makes Uploading API request and returns response as `dict`.

    It takes settings from `conf` module.

    Make sure that given `path` does not contain leading slash.

    Usage example:

```
>>> file_obj = open('photo.jpg', 'rb')
>>> uploading_request('POST', 'base/', files={'file': file_obj})
{
    'file': '9b9f4483-77b8-40ae-a198-272ba6280004'
}
>>> File('9b9f4483-77b8-40ae-a198-272ba6280004')
```

6.1.6 Exceptions

exception pyuploadcare.exceptions.**APIConnectionError**(*data=u''*, **args*, ***kwargs*)

    Network communication with Uploadcare errors.

exception pyuploadcare.exceptions.**APIError**(*data=u''*, **args*, ***kwargs*)

    API errors, e.g. bad json.

exception pyuploadcare.exceptions.**AuthenticationError**(*data=u''*, **args*, ***kwargs*)

    Authentication with Uploadcare's API errors.

exception pyuploadcare.exceptions.**InvalidParamError**(*data=u''*, **args*, ***kwargs*)

    Invalid parameters errors, e.g. invalid UUID

exception pyuploadcare.exceptions.**InvalidRequestError**(*data=u''*, **args*, ***kwargs*)

    Invalid service parameters errors, e.g status 404

exception pyuploadcare.exceptions.**ThrottledRequestError**(*response*)

    Raised when request was throttled.

exception pyuploadcare.exceptions.**TimeoutError**(*data=u''*, **args*, ***kwargs*)

    Timed out errors.

    It raises when user wants to wait the result of api requests, e.g.:

```
$ ucare store --wait 6c5e9526-b0fe-4739-8975-72e8d5ee6342
```

exception pyuploadcare.exceptions.**UploadError**(*data=u''*, **args*, ***kwargs*)

    Upload errors.

    It raises when user wants to wait the result of:

```
$ ucare upload_from_url --wait http://path.to/file.jpg
```

exception pyuploadcare.exceptions.**UploadcareException**(*data=u''*, **args*, ***kwargs*)

    Base exception class of library.

6.2 Django Widget API

6.2.1 Model Fields

class pyuploadcare.dj.models.**FileField**(*args*, **kwargs*)
> Django model field that stores uploaded file as Uploadcare CDN url.

class pyuploadcare.dj.models.**ImageField**(*manual_crop=None*, *args*, **kwargs*)
> Django model field that stores uploaded image as Uploadcare CDN url.
>
> It supports manual crop as well. *manual_crop* can be set to one of the following values:
>
> > • None, "disabled" — crop disabled;
> >
> > • "" — crop is enabled and the user will be able to select any area on an image;
> >
> > • "2:3" — user will be able to select an area with aspect ratio *2:3*;
> >
> > • "200x300" — same as previous, but if the selected area is bigger than *200x300*, it will be scaled down to these dimensions;
> >
> > • "200x300 upscale" — same as previous, but the selected area will be scaled even if it is smaller than the specified size.

class pyuploadcare.dj.models.**FileGroupField**(*args*, **kwargs*)
> Django model field that stores uploaded file group as Uploadcare CDN url.
>
> It provides multiple file uploading.

class pyuploadcare.dj.models.**ImageGroupField**(*args*, **kwargs*)
> Django model field that stores uploaded image group as Uploadcare CDN url.
>
> It provides multiple image uploading.

6.2.2 Form Fields

class pyuploadcare.dj.forms.**FileWidget**(*attrs=None*)
> Django form widget that sets up Uploadcare Widget.
>
> It adds js and hidden input with basic Widget's params, e.g. *data-public-key*.

class pyuploadcare.dj.forms.**FileField**(*args*, **kwargs*)
> Django form field that uses FileWidget with default arguments.
>
> It always returns URL.

class pyuploadcare.dj.forms.**ImageField**(*manual_crop=None*, *args*, **kwargs*)
> Django form field that sets up FileWidget to work with images.

class pyuploadcare.dj.forms.**FileGroupField**(*args*, **kwargs*)
> Django form field that sets up FileWidget in multiupload mode.

class pyuploadcare.dj.forms.**ImageGroupField**(*args*, **kwargs*)
> Django form field that sets up FileWidget in image multiupload mode.

6.3 Command Line Tool API

pyuploadcare.ucare_cli.**bar**(*iter_content*, *parts*, *title=u''*)

pyuploadcare.ucare_cli.**bool_or_none**(*value*)

pyuploadcare.ucare_cli.**build_filepath**(*path*, *file_*)

pyuploadcare.ucare_cli.**create_group**(*arg_namespace*)

pyuploadcare.ucare_cli.**delete_files**(*arg_namespace*)

pyuploadcare.ucare_cli.**get_file**(*arg_namespace*)

pyuploadcare.ucare_cli.**int_or_none**(*value*)

pyuploadcare.ucare_cli.**list_files**(*arg_namespace*)

pyuploadcare.ucare_cli.**load_config_from_args**(*arg_namespace*)

pyuploadcare.ucare_cli.**load_config_from_file**(*filename*)

pyuploadcare.ucare_cli.**main**(*arg_namespace=None*, *config_file_names=(u'~/.uploadcare'*, *u'uploadcare.ini')*)

pyuploadcare.ucare_cli.**pprint**(*value*)

pyuploadcare.ucare_cli.**save_file_locally**(*fname*, *response*, *size*)

pyuploadcare.ucare_cli.**store_files**(*arg_namespace*)

pyuploadcare.ucare_cli.**sync_files**(*arg_namespace*)

pyuploadcare.ucare_cli.**ucare_argparser**()

pyuploadcare.ucare_cli.**upload**(*arg_namespace*)

pyuploadcare.ucare_cli.**upload_from_url**(*arg_namespace*)

Indices and tables

- genindex
- modindex
- search

p

A

B

C

D

F

G

I