YaDisk Documentation

Release 1.3.0

Ivan Konovalov

Jul 06, 2019

Contents:

CHAPTER 1

Introduction

YaDisk-async is a modified version of YaDisk with `async/await` support. It uses *aiohttp* instead of *requests*.

The usage is more or less the same, except that you have to manually close all the sessions (can be done with *YaDisk.close* or through `async with` statement).

1.1 Installation

```
pip install yadisk-async
```

or

```
python setup.py install
```

1.2 Examples

```
import yadisk_async

y = yadisk_async.YaDisk(token="<token>")
# or
# y = yadisk_async.YaDisk("<application-id>", "<application-secret>", "<token>")

# Check if the token is valid
print(await y.check_token())

# Get disk information
print(await y.get_disk_info())

# Print files and directories at "/some/path"
print([i async for i in await y.listdir("/some/path")])
```

(continues on next page)

(continued from previous page)

```
# Upload "file_to_upload.txt" to "/destination.txt"
await y.upload("file_to_upload.txt", "/destination.txt")

# Same thing
with open("file_to_upload.txt", "rb") as f:
    await y.upload(f, "/destination.txt")

# Download "/some-file-to-download.txt" to "downloaded.txt"
await y.download("/some-file-to-download.txt", "downloaded.txt")

# Permanently remove "/file-to-remove"
await y.remove("/file-to-remove", permanently=True)

# Create a new directory at "/test-dir"
print(await y.mkdir("/test-dir"))

# Always remember to close all the connections or you'll get a warning
await y.close()
```

1.2.1 Receiving token with confirmation code

```
import asyncio
import sys
import yadisk_async

async def main():
    async with yadisk_async.YaDisk("application-id>", "<application-secret>") as y:
        url = y.get_code_url()

        print("Go to the following url: %s" % url)
        code = input("Enter the confirmation code: ")

        try:
            response = await y.get_token(code)
        except yadisk_async.exceptions.BadRequestError:
            print("Bad code")
            sys.exit(1)

        y.token = response.access_token

        if await y.check_token():
            print("Sucessfully received token!")
        else:
            print("Something went wrong. Not sure how though...")

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```

1.2.2 Recursive upload

```
import asyncio
import posixpath
```

(continues on next page)

(continued from previous page)

```
import os
import yadisk_async

def recursive_upload(from_dir, to_dir, n_parallel_requests=5):
    loop = asyncio.get_event_loop()

    y = yadisk_async.YaDisk(token="<application-token>")

    try:
        async def upload_files(queue):
            while queue:
                in_path, out_path = queue.pop(0)

                print("Uploading %s -> %s" % (in_path, out_path))

                try:
                    await y.upload(in_path, out_path)
                except yadisk_async.exceptions.PathExistsError:
                    print("%s already exists" % (out_path,))

        async def create_dirs(queue):
            while queue:
                path = queue.pop(0)

                print("Creating directory %s" % (path,))

                try:
                    await y.mkdir(path)
                except yadisk_async.exceptions.PathExistsError:
                    print("%s already exists" % (path,))

        mkdir_queue = []
        upload_queue = []

        print("Creating directory %s" % (to_dir,))

        try:
            loop.run_until_complete(y.mkdir(to_dir))
        except yadisk_async.exceptions.PathExistsError:
            print("%s already exists" % (to_dir,))

        for root, dirs, files in os.walk(from_dir):
            rel_dir_path = root.split(from_dir)[1].strip(os.path.sep)
            rel_dir_path = rel_dir_path.replace(os.path.sep, "/")
            dir_path = posixpath.join(to_dir, rel_dir_path)

            for dirname in dirs:
                mkdir_queue.append(posixpath.join(dir_path, dirname))

            for filename in files:
                out_path = posixpath.join(dir_path, filename)
                rel_dir_path_sys = rel_dir_path.replace("/", os.path.sep)
                in_path = os.path.join(from_dir, rel_dir_path_sys, filename)

                upload_queue.append((in_path, out_path))

            tasks = [upload_files(upload_queue) for i in range(n_parallel_requests)]
```

(continues on next page)

(continued from previous page)

```
            tasks.extend(create_dirs(mkdir_queue) for i in range(n_parallel_requests))

            loop.run_until_complete(asyncio.gather(*tasks))
    finally:
        loop.run_until_complete(y.close())

from_dir = input("Directory to upload: ")
to_dir = input("Destination directory: ")

recursive_upload(from_dir, to_dir, 5)
```

1.2.3 Setting custom properties of files

```
import asyncio
import yadisk_async

async def main():
    async with yadisk_async.YaDisk(token="<application-token>") as y:
        path = input("Enter a path to patch: ")
        properties = {"speed_of_light":       299792458,
                      "speed_of_light_units": "meters per second",
                      "message_for_owner":    "MWAHAHA! Your file has been patched by␣
↪an evil script!"}

        meta = await y.patch(path, properties)
        print("\nNew properties: ")

        for k, v in meta.custom_properties.items():
            print("%s: %r" % (k, v))

        answer = input("\nWant to get rid of them? (y/[n]) ")

        if answer.lower() in ("y", "yes"):
            properties = {k: None for k in properties}
            await y.patch(path, properties)
            print("Everything's back as usual")

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```

1.2.4 Emptying the trash bin

```
import asyncio
import sys
import yadisk_async

async def main():
    async with yadisk_async.YaDisk(token="<application-token>") as y:
        answer = input("Are you sure about this? (y/[n]) ")

        if answer.lower() in ("y", "yes"):
            print("Emptying the trash bin...")
```

(continues on next page)

(continued from previous page)

```
            operation = await y.remove_trash("/")
            print("It might take a while...")

            if operation is None:
                print("Nevermind. The deed is done.")
                sys.exit(0)

            while True:
                status = await y.get_operation_status(operation.href)

                if status == "in-progress":
                    await asyncio.sleep(5)
                    print("Still waiting...")
                elif status == "success":
                    print("Success!")
                    break
                else:
                    print("Got some weird status: %r" % (status,))
                    print("That's not normal")
                    break
        else:
            print("Not going to do anything")

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```

CHAPTER 2

Documentation

class yadisk_async.**YaDisk**(*id=''*, *secret=''*, *token=''*)

Implements access to Yandex.Disk REST API.

Note: Do not forget to call `YaDisk.close` or use the *async with* statement to close all the connections. Otherwise, you may get a warning.

In the original library this is handled in the destructor, but since `aiohttp.ClientSession.close` is a coroutine function the same cannot be done here, so you have to do it explicitly.

Parameters

- **id** – application ID
- **secret** – application secret password
- **token** – application token

Variables

- **id** – *str*, application ID
- **secret** – *str*, application secret password
- **token** – *str*, application token

check_token(*token=None*, ***kwargs*)

Check whether the token is valid.

Parameters

- **token** – token to check, equivalent to *self.token* if *None*
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries

- **retry_interval** – delay between retries in seconds

Returns *bool*

clear_session_cache()
Clears the session cache. Unused sessions will NOT be closed.

This method is not a coroutine.

close()
Closes all sessions and clears the session cache. Do not call this method while there are other active threads using this object.

This method can also be called implicitly by using the *async with* statement.

copy(*src_path*, *dst_path*, ***kwargs*)
Copy *src_path* to *dst_path*. If the operation is performed asynchronously, returns the link to the operation, otherwise, returns the link to the newly created resource.

Parameters

- **src_path** – source path
- **dst_path** – destination path
- **overwrite** – if *True* the destination path can be overwritten, otherwise, an error will be raised
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *LinkObject* or *OperationLinkObject*

download(*src_path*, *path_or_file*, ***kwargs*)
Download the file.

Parameters

- **src_path** – source path
- **path_or_file** – destination path or file-like object
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

download_public(*public_key*, *file_or_path*, ***kwargs*)
Download the public resource.

Parameters

- **public_key** – public key or public URL of the resource
- **file_or_path** – destination path or file-like object
- **path** – relative path to the resource within the public folder

- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

exists (*path*, ***kwargs*)
Check whether *path* exists.

Parameters

- **path** – path to the resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *bool*

get_auth_url (***kwargs*)
Get authentication URL for the user to go to.

This method is not a coroutine.

Parameters

- **type** – response type ("code" to get the confirmation code or "token" to get the token automatically)
- **device_id** – unique device ID, must be between 6 and 50 characters
- **device_name** – device name, should not be longer than 100 characters
- **display** – indicates whether to use lightweight layout, values other than "popup" are ignored
- **login_hint** – username or email for the account the token is being requested for
- **scope** – list of permissions for the application
- **optional_scope** – list of optional permissions for the application
- **force_confirm** – if True, user will be required to confirm access to the account even if the user has already granted access for the application
- **state** – The state string, which Yandex.OAuth returns without any changes (<= 1024 characters)

Returns authentication URL

get_code_url (***kwargs*)
Get the URL for the user to get the confirmation code. The confirmation code can later be used to get the token.

This method is not a coroutine.

Parameters

- **device_id** – unique device ID, must be between 6 and 50 characters
- **device_name** – device name, should not be longer than 100 characters

- **display** – indicates whether to use lightweight layout, values other than "popup" are ignored
- **login_hint** – username or email for the account the token is being requested for
- **scope** – list of permissions for the application
- **optional_scope** – list of optional permissions for the application
- **force_confirm** – if True, user will be required to confirm access to the account even if the user has already granted access for the application
- **state** – The state string, which Yandex.OAuth returns without any changes (<= 1024 characters)

Returns authentication URL

get_disk_info(***kwargs*)
Get disk information.

Parameters

- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *DiskInfoObject*

get_download_link(*path*, ***kwargs*)
Get a download link for a file (or a directory).

Parameters

- **path** – path to the resource
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *str*

get_files(***kwargs*)
Get a flat list of all files (that doesn't include directories).

Parameters

- **offset** – offset from the beginning of the list
- **limit** – number of list elements to be included
- **media_type** – type of files to include in the list
- **sort** – *str*, field to be used as a key to sort children resources
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*

- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns generator of `ResourceObject`

get_last_uploaded(***kwargs*)
Get the list of latest uploaded files sorted by upload date.

Parameters

- **limit** – maximum number of elements in the list
- **media_type** – type of files to include in the list
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns generator of `LastUploadedResourceListObject`

get_meta(*path*, ***kwargs*)
Get meta information about a file/directory.

Parameters

- **path** – path to the resource
- **limit** – number of children resources to be included in the response
- **offset** – number of children resources to be skipped in the response
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **sort** – *str*, field to be used as a key to sort children resources
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns `ResourceObject`

get_operation_status(*operation_id*, ***kwargs*)
Get operation status.

Parameters

- **operation_id** – ID of the operation or a link
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *str*

get_public_download_link(*public_key*, ***kwargs*)
Get a download link for a public resource.

Parameters

- **public_key** – public key or public URL of the resource
- **path** – relative path to the resource within the public folder
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *str*

get_public_meta(*public_key*, ***kwargs*)
Get meta-information about a public resource.

Parameters

- **public_key** – public key or public URL of the resource
- **path** – relative path to a resource in a public folder. By specifying the key of the published folder in *public_key*, you can request metainformation for any resource in the folder.
- **offset** – offset from the beginning of the list of nested resources
- **limit** – maximum number of nested elements to be included in the list
- **sort** – *str*, field to be used as a key to sort children resources
- **preview_size** – file preview size
- **preview_crop** – *bool*, allow preview crop
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *PublicResourceObject*

get_public_resources(***kwargs*)
Get a list of public resources.

Parameters

- **offset** – offset from the beginning of the list
- **limit** – maximum number of elements in the list
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **type** – filter based on type of resources ("file" or "dir")
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *PublicResourcesListObject*

get_public_type(*public_key*, ***kwargs*)
Get public resource type.

Parameters

- **public_key** – public key or public URL of the resource
- **path** – relative path to the resource within the public folder
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns "file" or "dir"

get_session(*token=None*)
Like *YaDisk.make_session* but cached.

This method is not a coroutine.

Returns *yadisk_async.session.SessionWithHeaders*, different instances for different threads

get_token(*code*, ***kwargs*)
Get a new token.

Parameters

- **code** – confirmation code
- **device_id** – unique device ID (between 6 and 50 characters)
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *TokenObject*

get_trash_meta(*path*, ***kwargs*)
Get meta information about a trash resource.

Parameters

- **path** – path to the trash resource
- **limit** – number of children resources to be included in the response
- **offset** – number of children resources to be skipped in the response
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **sort** – *str*, field to be used as a key to sort children resources
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *TrashResourceObject*

get_trash_type(*path*, ***kwargs*)
Get trash resource type.

Parameters

- **path** – path to the trash resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns "file" or "dir"

get_type(*path*, ***kwargs*)
Get resource type.

Parameters

- **path** – path to the resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns "file" or "dir"

get_upload_link(*path*, ***kwargs*)
Get a link to upload the file using the PUT request.

Parameters

- **path** – destination path

- **overwrite** – *bool*, determines whether to overwrite the destination
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *str*

is_dir(*path*, ***kwargs*)
Check whether *path* is a directory.

Parameters

- **path** – path to the resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *True* if *path* is a directory, *False* otherwise (even if it doesn't exist)

is_file(*path*, ***kwargs*)
Check whether *path* is a file.

Parameters

- **path** – path to the resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *True* if *path* is a file, *False* otherwise (even if it doesn't exist)

is_public_dir(*public_key*, ***kwargs*)
Check whether *public_key* is a public directory.

Parameters

- **public_key** – public key or public URL of the resource
- **path** – relative path to the resource within the public folder
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *True* if *public_key* is a directory, *False* otherwise (even if it doesn't exist)

is_public_file(*public_key*, ***kwargs*)
Check whether *public_key* is a public file.

Parameters

- **public_key** – public key or public URL of the resource
- **path** – relative path to the resource within the public folder
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *True* if *public_key* is a file, *False* otherwise (even if it doesn't exist)

is_trash_dir(*path*, ***kwargs*)
Check whether *path* is a trash directory.

Parameters

- **path** – path to the trash resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *True* if *path* is a directory, *False* otherwise (even if it doesn't exist)

is_trash_file(*path*, ***kwargs*)
Check whether *path* is a trash file.

Parameters

- **path** – path to the trash resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *True* if *path* is a directory, *False* otherwise (even if it doesn't exist)

listdir(*path*, ***kwargs*)
Get contents of *path*.

Parameters

- **path** – path to the directory
- **limit** – number of children resources to be included in the response
- **offset** – number of children resources to be skipped in the response
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers

- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns generator of *ResourceObject*

make_session(*token=None*)
Prepares *yadisk_async.session.SessionWithHeaders* object with headers needed for API.

This method is not a coroutine.

Parameters **token** – application token, equivalent to *self.token* if *None*

Returns *yadisk_async.session.SessionWithHeaders*

mkdir(*path*, ***kwargs*)
Create a new directory.

Parameters

- **path** – path to the directory to be created
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *LinkObject*

move(*src_path*, *dst_path*, ***kwargs*)
Move *src_path* to *dst_path*.

Parameters

- **src_path** – source path to be moved
- **dst_path** – destination path
- **overwrite** – *bool*, determines whether to overwrite the destination
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *OperationLinkObject* or *LinkObject*

patch(*path*, *properties*, ***kwargs*)
Update custom properties of a resource.

Parameters

- **path** – path to the resource
- **properties** – *dict*, custom properties to update
- **fields** – list of keys to be included in the response

- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns `ResourceObject`

public_exists(*public_key*, ***kwargs*)
Check whether the public resource exists.

Parameters

- **public_key** – public key or public URL of the resource
- **path** – relative path to the resource within the public folder
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *bool*

public_listdir(*public_key*, ***kwargs*)
Get contents of a public directory.

Parameters

- **public_key** – public key or public URL of the resource
- **path** – relative path to the resource in the public folder. By specifying the key of the published folder in *public_key*, you can request contents of any nested folder.
- **limit** – number of children resources to be included in the response
- **offset** – number of children resources to be skipped in the response
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns generator of `PublicResourceObject`

publish(*path*, ***kwargs*)
Make a resource public.

Parameters

- **path** – path to the resource to be published
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout

- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns `LinkObject`, link to the resource

refresh_token(*refresh_token*, ***kwargs*)
Refresh an existing token.

Parameters

- **refresh_token** – the refresh token that was received with the token
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns `TokenObject`

remove(*path*, ***kwargs*)
Remove the resource.

Parameters

- **path** – path to the resource to be removed
- **permanently** – if *True*, the resource will be removed permanently, otherwise, it will be just moved to the trash
- **md5** – *str*, MD5 hash of the file to remove
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns `OperationLinkObject` if the operation is performed asynchronously, *None* otherwise

remove_trash(*path*, ***kwargs*)
Remove a trash resource.

Parameters

- **path** – path to the trash resource to be deleted
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns `OperationLinkObject` if the operation is performed asynchronously, *None* otherwise

restore_trash(*path*, *dst_path=None*, ***kwargs*)
Restore a trash resource. Returns a link to the newly created resource or a link to the asynchronous operation.

Parameters

- **path** – path to the trash resource to restore
- **dst_path** – destination path
- **overwrite** – *bool*, determines whether the destination can be overwritten
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns `LinkObject` or `OperationLinkObject`

revoke_token(*token=None*, ***kwargs*)
Revoke the token.

Parameters

- **token** – token to revoke, equivalent to *self.token* if *None*
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns `TokenRevokeStatusObject`

save_to_disk(*public_key*, ***kwargs*)
Saves a public resource to the disk. Returns the link to the operation if it's performed asynchronously, or a link to the resource otherwise.

Parameters

- **public_key** – public key or public URL of the resource
- **name** – filename of the saved resource
- **path** – path to the copied resource in the public folder
- **save_path** – path to the destination directory (downloads directory by default)
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries

- **retry_interval** – delay between retries in seconds

Returns *LinkObject* or *OperationLinkObject*

trash_exists(*path*, ***kwargs*)
Check whether the trash resource at *path* exists.

Parameters

- **path** – path to the trash resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *bool*

trash_listdir(*path*, ***kwargs*)
Get contents of a trash resource.

Parameters

- **path** – path to the directory in the trash bin
- **limit** – number of children resources to be included in the response
- **offset** – number of children resources to be skipped in the response
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns generator of *TrashResourceObject*

unpublish(*path*, ***kwargs*)
Make a public resource private.

Parameters

- **path** – path to the resource to be unpublished
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *LinkObject*, link to the resource

upload(*path_or_file*, *dst_path*, ***kwargs*)
Upload a file to disk.

Parameters

- **path_or_file** – path or file-like object to be uploaded
- **dst_path** – destination path
- **overwrite** – if *True*, the resource will be overwritten if it already exists, an error will be raised otherwise
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

upload_url(*url*, *path*, ***kwargs*)
Upload a file from URL.

Parameters

- **url** – source URL
- **path** – destination path
- **disable_redirects** – *bool*, forbid redirects
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns `OperationLinkObject`, link to the asynchronous operation

2.1 General parameters

Almost all methods of *YaDisk* (the ones that accept ***kwargs*) accept some additional arguments:

- **n_retries** - *int*, maximum number of retries for a request
- **retry_delay** - *float*, delay between retries (in seconds)
- **headers** - *dict* or *None*, additional request headers

aiohttp parameters like *timeout*, *proxies*, etc. are also accepted (see `aiohttp.request()`).

This also applies to low-level functions and API request objects as well.

2.2 Settings

The following settings can be accessed and changed at runtime in *yadisk_async.settings* module:

- **DEFAULT_TIMEOUT** - `aiohttp.ClientTimeout`, default timeout for requests.
- **DEFAULT_N_RETRIES** - *int*, default number of retries

- **DEFAULT_RETRY_INTERVAL** - *float*, default retry interval
- **DEFAULT_UPLOAD_TIMEOUT** - analogous to *DEFAULT_TIMEOUT* but for *upload* function
- **DEFAULT_UPLOAD_RETRY_INTERVAL** - analogous to *DEFAULT_RETRY_INTERVAL* but for *upload* function

2.3 Exceptions

Aside from the exceptions listed below, API requests can also raise exceptions in *aiohttp*.

exception yadisk_async.exceptions.**YaDiskError**(*error_type=None*, *msg=''*, *response=None*)

Bases: Exception

Base class for all exceptions in this library.

Variables

- **error_type** – *str*, unique error code as returned by API
- **response** – an instance of aiohttp.ClientResponse

Parameters

- **error_type** – *str*, unique error code as returned by API
- **msg** – *str*, exception message
- **response** – an instance of aiohttp.ClientResponse

exception yadisk_async.exceptions.**RetriableYaDiskError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.YaDiskError*

Thrown when there was an error but it would make sense to retry the request.

exception yadisk_async.exceptions.**UnknownYaDiskError**(*msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.RetriableYaDiskError*

Thrown when the request failed but the response does not contain any error info.

exception yadisk_async.exceptions.**WrongResourceTypeError**(*msg=''*)

Bases: *yadisk_async.exceptions.YaDiskError*

Thrown when the resource was expected to be of different type (e.g., file instead of directory).

exception yadisk_async.exceptions.**BadRequestError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.YaDiskError*

Thrown when the server returns code 400.

exception yadisk_async.exceptions.**UnauthorizedError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.YaDiskError*

Thrown when the server returns code 401.

exception yadisk_async.exceptions.**ForbiddenError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.YaDiskError*

Thrown when the server returns code 403.

exception yadisk_async.exceptions.**NotFoundError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.YaDiskError*

Thrown when the server returns code 404.

exception yadisk_async.exceptions.**NotAcceptableError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.YaDiskError*

Thrown when the server returns code 406.

exception yadisk_async.exceptions.**ConflictError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.YaDiskError*

Thrown when the server returns code 409.

exception yadisk_async.exceptions.**UnsupportedMediaError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.YaDiskError*

Thrown when the server returns code 415.

exception yadisk_async.exceptions.**LockedError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.YaDiskError*

Thrown when the server returns code 423.

exception yadisk_async.exceptions.**TooManyRequestsError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.YaDiskError*

Thrown when the server returns code 429.

exception yadisk_async.exceptions.**InternalServerError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.RetriableYaDiskError*

Thrown when the server returns code 500.

exception yadisk_async.exceptions.**BadGatewayError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.RetriableYaDiskError*

Thrown when the server returns code 502

exception yadisk_async.exceptions.**UnavailableError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.RetriableYaDiskError*

Thrown when the server returns code 503.

exception yadisk_async.exceptions.**GatewayTimeoutError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.RetriableYaDiskError*

Thrown when the server returns code 504

exception yadisk_async.exceptions.**InsufficientStorageError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.YaDiskError*

Thrown when the server returns code 509.

exception yadisk_async.exceptions.**PathNotFoundError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.NotFoundError*

Thrown when the requested path does not exist.

exception yadisk_async.exceptions.**ParentNotFoundError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.ConflictError*

Thrown by *mkdir*, *upload*, etc. when the parent directory doesn't exist.

exception yadisk_async.exceptions.**PathExistsError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.ConflictError*

Thrown when the requested path already exists.

exception yadisk_async.exceptions.**DirectoryExistsError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.PathExistsError*

Thrown when the directory already exists.

exception yadisk_async.exceptions.**FieldValidationError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.BadRequestError*

Thrown when the request contains fields with invalid data.

exception yadisk_async.exceptions.**ResourceIsLockedError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.LockedError*

Thrown when the resource is locked by another operation.

exception yadisk_async.exceptions.**MD5DifferError**(*error_type=None*, *msg=''*, *response=None*)

Bases: *yadisk_async.exceptions.ConflictError*

Thrown when the MD5 hash of the file to be deleted doesn't match with the actual one.

2.4 Objects

class yadisk_async.objects.**YaDiskObject**(*field_types=None*)

Base class for all objects mirroring the ones returned by Yandex.Disk REST API. It must have a fixed number of fields, each field must have a type. It also supports subscripting and access of fields through the . operator.

Parameters **field_types** – *dict* or *None*

import_fields(*source_dict*)

Set all the fields of the object to the values in *source_dict*. All the other fields are ignored

Parameters **source_dict** – *dict* or *None* (nothing will be done in that case)

remove_alias(*alias*)

Remove an alias.

Parameters **alias** – *str*

remove_field(*field*)

Remove field.

Parameters **field** – *str*

set_alias(*alias*, *name*)
Set an alias.

Parameters

- **alias** – *str*, alias to add
- **name** – *str*, field name

set_field_type(*field*, *type*)
Set field type.

Parameters

- **field** – *str*
- **type** – type or factory

set_field_types(*field_types*)
Set the field types of the object

Parameters **field_types** – *dict*, where keys are the field names and values are types (or factories)

class yadisk_async.objects.**ErrorObject**(*error=None*)
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

Mirrors Yandex.Disk REST API error object.

Parameters **error** – *dict* or *None*

Variables

- **message** – *str*, human-readable error message
- **description** – *str*, technical error description
- **error** – *str*, error code

class yadisk_async.objects.auth.**TokenObject**(*token=None*)
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

Token object.

Parameters **token** – *dict* or *None*

Variables

- **access_token** – *str*, token string
- **refresh_token** – *str*, the refresh-token
- **token_type** – *str*, type of the token
- **expires_in** – *int*, amount of time before the token expires

class yadisk_async.objects.auth.**TokenRevokeStatusObject**(*token_revoke_status=None*)
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

Result of token revocation request.

Parameters **token_revoke_status** – *dict* or *None*

Variables **status** – *str*, status of the operation

class yadisk_async.objects.disk.**DiskInfoObject**(*disk_info=None*)
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

Disk information object.

Parameters **disk_info** – *dict* or *None*

Variables

- **max_file_size** – *int*, maximum supported file size (bytes)
- **unlimited_autoupload_enabled** – *bool*, tells whether unlimited autoupload from mobile devices is enabled
- **total_space** – *int*, total disk size (bytes)
- **trash_size** – *int*, amount of space used by trash (bytes), part of *used_space*
- **is_paid** – *bool*, tells if the account is paid or not
- **used_space** – *int*, amount of space used (bytes)
- **system_folders** – *SystemFoldersObject*, paths to the system folders
- **user** – *UserObject*, owner of the disk
- **revision** – *int*, current revision of Yandex.Disk

class yadisk_async.objects.disk.**SystemFoldersObject**(*system_folders=None*)

Bases: yadisk_async.objects.yadisk_object.YaDiskObject

Object, containing paths to system folders.

Parameters **system_folders** – *dict* or *None*

Variables

- **odnoklassniki** – *str*, path to the Odnoklassniki folder
- **google** – *str*, path to the Google+ folder
- **instagram** – *str*, path to the Instagram folder
- **vkontakte** – *str*, path to the VKontakte folder
- **mailru** – *str*, path to the My World folder
- **facebook** – *str*, path to the Facebook folder
- **social** – *str*, path to the social networks folder
- **screenshots** – *str*, path to the screenshot folder
- **photostream** – *str*, path to the camera folder

class yadisk_async.objects.disk.**UserObject**(*user=None*)

Bases: yadisk_async.objects.yadisk_object.YaDiskObject

User object.

Parameters **user** – *dict* or *None*

Variables

- **country** – *str*, user's country
- **login** – *str*, user's login
- **display_name** – *str*, user's display name
- **uid** – *str*, user's UID

class yadisk_async.objects.disk.**UserPublicInfoObject**(*public_user_info=None*)
Bases: *yadisk_async.objects.disk.UserObject*

Public user information object. Inherits from *UserObject* for compatibility.

Parameters **public_user_info** – *dict* or *None*

Variables

- **login** – *str*, user's login
- **display_name** – *str*, user's display name
- **uid** – *str*, user's UID

class yadisk_async.objects.resources.**CommentIDsObject**(*comment_ids=None*)
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

Comment IDs object.

Parameters **comment_ids** – *dict* or *None*

Variables

- **private_resource** – *str*, comment ID for private resources
- **public_resource** – *str*, comment ID for public resources

class yadisk_async.objects.resources.**EXIFObject**(*exif=None*)
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

EXIF metadata object.

Parameters **exif** – *dict* or *None*

Variables **date_time** – datetime.datetime, capture date

class yadisk_async.objects.resources.**FilesResourceListObject**(*files_resource_list=None*)
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

Flat list of files.

Parameters **files_resource_list** – *dict* or *None*

Variables

- **items** – *list*, flat list of files (*ResourceObject*)
- **limit** – *int*, maximum number of elements in the list
- **offset** – *int*, offset from the beginning of the list

class yadisk_async.objects.resources.**LastUploadedResourceListObject**(*last_uploaded_resources_list=No*
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

List of last uploaded resources.

Parameters **last_uploaded_resources_list** – *dict* or *None*

Variables

- **items** – *list*, list of resources (*ResourceObject*)
- **limit** – *int*, maximum number of elements in the list

class yadisk_async.objects.resources.**LinkObject**(*link=None*)
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

Link object.

Parameters **link** – *dict* or *None*

Variables

- **href** – *str*, link URL
- **method** – *str*, HTTP method
- **templated** – *bool*, tells whether the URL is templated

class yadisk_async.objects.resources.**OperationLinkObject**(*link=None*)
Bases: *yadisk_async.objects.resources.LinkObject*

Operation link object.

Parameters **link** – *dict* or *None*

Variables

- **href** – *str*, link URL
- **method** – *str*, HTTP method
- **templated** – *bool*, tells whether the URL is templated

class yadisk_async.objects.resources.**PublicResourcesListObject**(*public_resources_list=None*)
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

List of public resources.

Parameters **public_resources_list** – *dict* or *None*

Variables

- **items** – *list*, list of public resources (*PublicResourceObject*)
- **type** – *str*, resource type to filter by
- **limit** – *int*, maximum number of elements in the list
- **offset** – *int*, offset from the beginning of the list

class yadisk_async.objects.resources.**ResourceListObject**(*resource_list=None*)
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

List of resources.

Parameters **resource_list** – *dict* or *None*

Variables

- **sort** – *str*, sort type
- **items** – *list*, list of resources (*ResourceObject*)
- **limit** – *int*, maximum number of elements in the list
- **offset** – *int*, offset from the beginning of the list
- **path** – *str*, path to the directory that contains the elements of the list
- **total** – *int*, number of elements in the list

class yadisk_async.objects.resources.**ResourceObject**(*resource=None*)
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

Resource object.

Parameters **resource** – *dict* or *None*

Variables

- **antivirus_status** – *str*, antivirus check status
- **file** – *str*, download URL
- **size** – *int*, file size
- **public_key** – *str*, public resource key
- **sha256** – *str*, SHA256 hash
- **md5** – *str*, MD5 hash
- **embedded** – *ResourceListObject*, list of nested resources
- **name** – *str*, filename
- **exif** – *EXIFObject*, EXIF metadata
- **resource_id** – *str*, resource ID
- **custom_properties** – *dict*, custom resource properties
- **public_url** – *str*, public URL
- **share** – *ShareInfoObject*, shared folder information
- **modified** – datetime.datetime, date of last modification
- **created** – datetime.datetime, date of creation
- **photoslice_time** – datetime.datetime, photo/video creation date
- **mime_type** – *str*, MIME type
- **path** – *str*, path to the resource
- **preview** – *str*, file preview URL
- **comment_ids** – *CommentIDsObject*, comment IDs
- **type** – *str*, type ("file" or "dir")
- **media_type** – *str*, file type as determined by Yandex.Disk
- **revision** – *int*, Yandex.Disk revision at the time of last modification

class yadisk_async.objects.resources.**ResourceUploadLinkObject**(*resource_upload_link=None*)

Bases: *yadisk_async.objects.resources.LinkObject*

Resource upload link.

Parameters **resource_upload_link** – *dict* or *None*

Variables

- **operation_id** – *str*, ID of the upload operation
- **href** – *str*, link URL
- **method** – *str*, HTTP method
- **templated** – *bool*, tells whether the URL is templated

class yadisk_async.objects.resources.**ShareInfoObject**(*share_info=None*)

Bases: yadisk_async.objects.yadisk_object.YaDiskObject

Shared folder information object.

Parameters **share_info** – *dict* or *None*

Variables

- **is_root** – *bool*, tells whether the folder is root
- **is_owned** – *bool*, tells whether the user is the owner of this directory
- **rights** – *str*, access rights

class yadisk_async.objects.resources.**PublicResourceObject**(*public_resource=None*)

Bases: *yadisk_async.objects.resources.ResourceObject*

Public resource object.

Parameters **resource** – *dict* or *None*

Variables

- **antivirus_status** – *str*, antivirus check status
- **file** – *str*, download URL
- **size** – *int*, file size
- **public_key** – *str*, public resource key
- **sha256** – *str*, SHA256 hash
- **md5** – *str*, MD5 hash
- **embedded** – *PublicResourceObject*, list of nested resources
- **name** – *str*, filename
- **exif** – *EXIFObject*, EXIF metadata
- **resource_id** – *str*, resource ID
- **custom_properties** – *dict*, custom resource properties
- **public_url** – *str*, public URL
- **share** – *ShareInfoObject*, shared folder information
- **modified** – datetime.datetime, date of last modification
- **created** – datetime.datetime, date of creation
- **photoslice_time** – datetime.datetime, photo/video creation date
- **mime_type** – *str*, MIME type
- **path** – *str*, path to the resource
- **preview** – *str*, file preview URL
- **comment_ids** – *CommentIDsObject*, comment IDs
- **type** – *str*, type (“file” or “dir”)
- **media_type** – *str*, file type as determined by Yandex.Disk
- **revision** – *int*, Yandex.Disk revision at the time of last modification
- **view_count** – *int*, number of times the public resource was viewed
- **owner** – *UserPublicInfoObject*, owner of the public resource

class yadisk_async.objects.resources.**PublicResourceListObject**(*public_resource_list=None*)

Bases: *yadisk_async.objects.resources.ResourceListObject*

List of public resources.

Parameters **public_resource_list** – *dict* or *None*

Variables

- **sort** – *str*, sort type
- **items** – *list*, list of resources (*ResourceObject*)
- **limit** – *int*, maximum number of elements in the list
- **offset** – *int*, offset from the beginning of the list
- **path** – *str*, path to the directory that contains the elements of the list
- **total** – *int*, number of elements in the list
- **public_key** – *str*, public key of the resource

class yadisk_async.objects.resources.**TrashResourceObject**(*trash_resource=None*)

Bases: *yadisk_async.objects.resources.ResourceObject*

Trash resource object.

Parameters **trash_resource** – *dict* or *None*

Variables

- **antivirus_status** – *str*, antivirus check status
- **file** – *str*, download URL
- **size** – *int*, file size
- **public_key** – *str*, public resource key
- **sha256** – *str*, SHA256 hash
- **md5** – *str*, MD5 hash
- **embedded** – *TrashResourceListObject*, list of nested resources
- **name** – *str*, filename
- **exif** – *EXIFObject*, EXIF metadata
- **resource_id** – *str*, resource ID
- **custom_properties** – *dict*, custom resource properties
- **public_url** – *str*, public URL
- **share** – *ShareInfoObject*, shared folder information
- **modified** – datetime.datetime, date of last modification
- **created** – datetime.datetime, date of creation
- **photoslice_time** – datetime.datetime, photo/video creation date
- **mime_type** – *str*, MIME type
- **path** – *str*, path to the resource
- **preview** – *str*, file preview URL
- **comment_ids** – *CommentIDsObject*, comment IDs
- **type** – *str*, type (“file” or “dir”)
- **media_type** – *str*, file type as determined by Yandex.Disk

- **revision** – *int*, Yandex.Disk revision at the time of last modification
- **origin_path** – *str*, original path
- **deleted** – `datetime.datetime`, date of deletion

class yadisk_async.objects.resources.**TrashResourceListObject**(*trash_resource_list=None*)
Bases: *yadisk_async.objects.resources.ResourceListObject*

List of trash resources.

Parameters **trash_resource_list** – *dict* or *None*

Variables

- **sort** – *str*, sort type
- **items** – *list*, list of resources (*TrashResourceObject*)
- **limit** – *int*, maximum number of elements in the list
- **offset** – *int*, offset from the beginning of the list
- **path** – *str*, path to the directory that contains the elements of the list
- **total** – *int*, number of elements in the list

class yadisk_async.objects.operations.**OperationStatusObject**(*operation_status=None*)
Bases: yadisk_async.objects.yadisk_object.YaDiskObject

Operation status object.

Parameters **operation_status** – *dict* or *None*

Variables

- **type** – *str*, type of the operation
- **status** – *str*, status of the operation
- **operation_id** – *str*, ID of the operation
- **link** – *LinkObject*, link to the operation
- **data** – *dict*, other information about the operation

2.5 Low-level API

2.5.1 Utilities

yadisk_async.utils.**get_exception**(*response*)
Get an exception instance based on response, assuming the request has failed.

Parameters **response** – an instance of `aiohttp.ClientResponse`

Returns an exception instance, subclass of *YaDiskError*

yadisk_async.utils.**auto_retry**(*func*, *n_retries=None*, *retry_interval=None*)
Attempt to perform a request with automatic retries. A retry is triggered by `aiohttp.ClientError` or *RetriableYaDiskError*.

Parameters

- **func** – function to run, must not require any arguments

- **n_retries** – *int*, maximum number of retries
- **retry_interval** – *int* or *float*, delay between retries (in seconds)

Returns return value of func()

2.5.2 Functions

yadisk_async.functions.auth.**check_token**(*session*, ***kwargs*)
Check whether the token is valid.

Parameters **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers

Returns *bool*

yadisk_async.functions.auth.**get_auth_url**(*client_id*, ***kwargs*)
Get authentication URL for the user to go to.

Parameters

- **client_id** – application ID
- **type** – response type ("code" to get the confirmation code or "token" to get the token automatically)
- **device_id** – unique device ID, must be between 6 and 50 characters
- **device_name** – device name, should not be longer than 100 characters
- **display** – indicates whether to use lightweight layout, values other than "popup" are ignored
- **login_hint** – username or email for the account the token is being requested for
- **scope** – list of permissions for the application
- **optional_scope** – list of optional permissions for the application
- **force_confirm** – if True, user will be required to confirm access to the account even if the user has already granted access for the application
- **state** – The state string, which Yandex.OAuth returns without any changes (<= 1024 characters)

Returns authentication URL

yadisk_async.functions.auth.**get_code_url**(*client_id*, ***kwargs*)
Get the URL for the user to get the confirmation code. The confirmation code can later be used to get the token.

Parameters

- **client_id** – application ID
- **device_id** – unique device ID, must be between 6 and 50 characters
- **device_name** – device name, should not be longer than 100 characters
- **display** – indicates whether to use lightweight layout, values other than "popup" are ignored
- **login_hint** – username or email for the account the token is being requested for
- **scope** – list of permissions for the application
- **optional_scope** – list of optional permissions for the application

- **force_confirm** – if True, user will be required to confirm access to the account even if the user has already granted access for the application
- **state** – The state string, which Yandex.OAuth returns without any changes (<= 1024 characters)

Returns authentication URL

yadisk_async.functions.auth.**get_token**(*code*, *client_id*, *client_secret*, ***kwargs*)

Get a new token.

Parameters

- **code** – confirmation code
- **client_id** – application ID
- **client_secret** – application secret password
- **device_id** – unique device ID (between 6 and 50 characters)
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *TokenObject*

yadisk_async.functions.auth.**refresh_token**(*refresh_token*, *client_id*, *client_secret*, ***kwargs*)

Refresh an existing token.

Parameters

- **refresh_token** – the refresh token that was received with the token
- **client_id** – application ID
- **client_secret** – application secret password
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *TokenObject*

yadisk_async.functions.auth.**revoke_token**(*token*, *client_id*, *client_secret*, ***kwargs*)

Revoke the token.

Parameters

- **token** – token to revoke
- **client_id** – application ID
- **client_secret** – application secret password
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries

- **retry_interval** – delay between retries in seconds

Returns *TokenRevokeStatusObject*

yadisk_async.functions.disk.**get_disk_info**(*session*, ***kwargs*)
Get disk information.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *DiskInfoObject*

yadisk_async.functions.resources.**copy**(*session*, *src_path*, *dst_path*, ***kwargs*)
Copy *src_path* to *dst_path*. If the operation is performed asynchronously, returns the link to the operation, otherwise, returns the link to the newly created resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **src_path** – source path
- **dst_path** – destination path
- **overwrite** – if *True* the destination path can be overwritten, otherwise, an error will be raised
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *LinkObject* or *OperationLinkObject*

yadisk_async.functions.resources.**download**(*session*, *src_path*, *file_or_path*, ***kwargs*)
Download the file.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **src_path** – source path
- **path_or_file** – destination path or file-like object
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers

- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

yadisk_async.functions.resources.**exists**(*session*, *path*, ***kwargs*)
Check whether *path* exists.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *bool*

yadisk_async.functions.resources.**get_download_link**(*session*, *path*, ***kwargs*)
Get a download link for a file (or a directory).

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *str*

yadisk_async.functions.resources.**get_meta**(*session*, *path*, ***kwargs*)
Get meta information about a file/directory.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource
- **limit** – number of children resources to be included in the response
- **offset** – number of children resources to be skipped in the response
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **sort** – *str*, field to be used as a key to sort children resources
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout

- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *ResourceObject*

yadisk_async.functions.resources.**get_type**(*session*, *path*, ***kwargs*)
Get resource type.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns "file" or "dir"

yadisk_async.functions.resources.**get_upload_link**(*session*, *path*, ***kwargs*)
Get a link to upload the file using the PUT request.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – destination path
- **overwrite** – *bool*, determines whether to overwrite the destination
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *str*

yadisk_async.functions.resources.**is_dir**(*session*, *path*, ***kwargs*)
Check whether *path* is a directory.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *True* if *path* is a directory, *False* otherwise (even if it doesn't exist)

yadisk_async.functions.resources.**is_file**(*session*, *path*, ***kwargs*)
Check whether *path* is a file.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *True* if *path* is a file, *False* otherwise (even if it doesn't exist)

yadisk_async.functions.resources.**listdir**(*session*, *path*, ***kwargs*)
Get contents of *path*.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the directory
- **limit** – number of children resources to be included in the response
- **offset** – number of children resources to be skipped in the response
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns generator of *ResourceObject*

yadisk_async.functions.resources.**mkdir**(*session*, *path*, ***kwargs*)
Create a new directory.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the directory to be created
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries

- **retry_interval** – delay between retries in seconds

Returns *LinkObject*

yadisk_async.functions.resources.**remove**(*session*, *path*, ***kwargs*)
Remove the resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource to be removed
- **permanently** – if *True*, the resource will be removed permanently, otherwise, it will be just moved to the trash
- **md5** – *str*, MD5 hash of the file to remove
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *LinkObject* if the operation is performed asynchronously, *None* otherwise

yadisk_async.functions.resources.**upload**(*session*, *file_or_path*, *dst_path*, ***kwargs*)
Upload a file to disk.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **file_or_path** – path or file-like object to be uploaded
- **dst_path** – destination path
- **overwrite** – if *True*, the resource will be overwritten if it already exists, an error will be raised otherwise
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

yadisk_async.functions.resources.**get_trash_meta**(*session*, *path*, ***kwargs*)
Get meta information about a trash resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the trash resource
- **limit** – number of children resources to be included in the response
- **offset** – number of children resources to be skipped in the response

- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **sort** – *str*, field to be used as a key to sort children resources
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *TrashResourceObject*

yadisk_async.functions.resources.**trash_exists**(*session*, *path*, ***kwargs*)

Check whether the trash resource at *path* exists.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the trash resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *bool*

yadisk_async.functions.resources.**restore_trash**(*session*, *path*, *dst_path=None*, ***kwargs*)

Restore a trash resource. Returns a link to the newly created resource or a link to the asynchronous operation.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the trash resource to be restored
- **dst_path** – destination path
- **overwrite** – *bool*, determines whether the destination can be overwritten
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *LinkObject* or *OperationLinkObject*

yadisk_async.functions.resources.**move**(*session*, *src_path*, *dst_path*, ***kwargs*)

Move *src_path* to *dst_path*.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **src_path** – source path to be moved
- **dst_path** – destination path
- **overwrite** – *bool*, determines whether to overwrite the destination
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *LinkObject* or *OperationLinkObject*

yadisk_async.functions.resources.**remove_trash**(*session*, *path*, ***kwargs*)
Remove a trash resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the trash resource to be deleted
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *OperationLinkObject* if the operation is performed asynchronously, *None* otherwise

yadisk_async.functions.resources.**publish**(*session*, *path*, ***kwargs*)
Make a resource public.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource to be published
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *LinkObject*, link to the resource

yadisk_async.functions.resources.**unpublish**(*session*, *path*, ***kwargs*)
Make a public resource private.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource to be unpublished
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *LinkObject*

yadisk_async.functions.resources.**save_to_disk**(*session*, *public_key*, ***kwargs*)
Saves a public resource to the disk. Returns the link to the operation if it's performed asynchronously, or a link to the resource otherwise.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **public_key** – public key or public URL of the public resource
- **name** – filename of the saved resource
- **path** – path to the copied resource in the public folder
- **save_path** – path to the destination directory (downloads directory by default)
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *LinkObject* or *OperationLinkObject*

yadisk_async.functions.resources.**get_public_meta**(*session*, *public_key*, ***kwargs*)
Get meta-information about a public resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **public_key** – public key or public URL of the public resource
- **path** – relative path to a resource in a public folder. By specifying the key of the published folder in *public_key*, you can request metainformation for any resource in the folder.
- **offset** – offset from the beginning of the list of nested resources
- **limit** – maximum number of nested elements to be included in the list

- **sort** – *str*, field to be used as a key to sort children resources
- **preview_size** – file preview size
- **preview_crop** – *bool*, allow preview crop
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *PublicResourceObject*

yadisk_async.functions.resources.**public_exists**(*session*, *public_key*, ***kwargs*)

Check whether the public resource exists.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **public_key** – public key or public URL of the public resource
- **path** – relative path to the resource within the public folder
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *bool*

yadisk_async.functions.resources.**public_listdir**(*session*, *public_key*, ***kwargs*)

Get contents of a public directory.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **public_key** – public key or public URL of the public resource
- **path** – relative path to the resource in the public folder. By specifying the key of the published folder in *public_key*, you can request contents of any nested folder.
- **limit** – number of children resources to be included in the response
- **offset** – number of children resources to be skipped in the response
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns generator of *PublicResourceObject*

yadisk_async.functions.resources.**get_public_type**(*session*, *public_key*, ***kwargs*)
Get public resource type.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **public_key** – public key or public URL of the public resource
- **path** – relative path to the resource within the public folder
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns "file" or "dir"

yadisk_async.functions.resources.**is_public_dir**(*session*, *public_key*, ***kwargs*)
Check whether the public resource is a public directory.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **public_key** – public key or public URL of the public resource
- **path** – relative path to the resource within the public folder
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *True* if *public_key* is a directory, *False* otherwise (even if it doesn't exist)

yadisk_async.functions.resources.**is_public_file**(*session*, *public_key*, ***kwargs*)
Check whether the public resource is a public file.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **public_key** – public key or public URL of the public resource
- **path** – relative path to the resource within the public folder
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *True* if *public_key* is a file, *False* otherwise (even if it doesn't exist)

yadisk_async.functions.resources.**trash_listdir**(*session*, *path*, ***kwargs*)
Get contents of a trash resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the directory in the trash bin
- **limit** – number of children resources to be included in the response
- **offset** – number of children resources to be skipped in the response
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns generator of *TrashResourceObject*

yadisk_async.functions.resources.**get_trash_type**(*session*, *path*, ***kwargs*)
Get trash resource type.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the trash resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns "file" or "dir"

yadisk_async.functions.resources.**is_trash_dir**(*session*, *path*, ***kwargs*)
Check whether *path* is a trash directory.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the trash resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *True* if *path* is a directory, *False* otherwise (even if it doesn't exist)

yadisk_async.functions.resources.**is_trash_file**(*session*, *path*, ***kwargs*)
Check whether *path* is a trash file.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the trash resource
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *True* if *path* is a file, *False* otherwise (even if it doesn't exist)

yadisk_async.functions.resources.**get_public_resources**(*session*, ***kwargs*)
Get a list of public resources.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **offset** – offset from the beginning of the list
- **limit** – maximum number of elements in the list
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **type** – filter based on type of resources ("file" or "dir")
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *PublicResourcesListObject*

yadisk_async.functions.resources.**patch**(*session*, *path*, *properties*, ***kwargs*)
Update custom properties of a resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource
- **properties** – *dict*, custom properties to update
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries

- **retry_interval** – delay between retries in seconds

Returns *ResourceObject*

yadisk_async.functions.resources.**get_files**(*session*, ***kwargs*)
Get a flat list of all files (that doesn't include directories).

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **offset** – offset from the beginning of the list
- **limit** – number of list elements to be included
- **media_type** – type of files to include in the list
- **sort** – *str*, field to be used as a key to sort children resources
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns generator of *ResourceObject*

yadisk_async.functions.resources.**get_last_uploaded**(*session*, ***kwargs*)
Get the list of latest uploaded files sorted by upload date.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **limit** – maximum number of elements in the list
- **media_type** – type of files to include in the list
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns generator of *LastUploadedResourceListObject*

yadisk_async.functions.resources.**upload_url**(*session*, *url*, *path*, ***kwargs*)
Upload a file from URL.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **url** – source URL
- **path** – destination path
- **disable_redirects** – *bool*, forbid redirects
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *OperationLinkObject*, link to the asynchronous operation

yadisk_async.functions.resources.**get_public_download_link**(*session*, *public_key*, ***kwargs*)

Get a download link for a public resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **public_key** – public key or public URL of the public resource
- **path** – relative path to the resource within the public folder
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *str*

yadisk_async.functions.resources.**download_public**(*session*, *public_key*, *file_or_path*, ***kwargs*)

Download the public resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **public_key** – public key or public URL of the public resource
- **file_or_path** – destination path or file-like object
- **path** – relative path to the resource within the public folder
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

yadisk_async.functions.operations.**get_operation_status**(*session*, *operation_id*, ***kwargs*)

Get operation status.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **operation_id** – ID of the operation or a link
- **fields** – list of keys to be included in the response
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds

Returns *str*

2.5.3 API request objects

class yadisk_async.api.**APIRequest**(*session*, *args*, ***kwargs*)

Base class for all API requests.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders*
- **args** – *dict* of arguments, that will be passed to *process_args*
- **timeout** – *float* or *tuple*, request timeout
- **headers** – *dict* or *None*, additional request headers
- **n_retries** – *int*, maximum number of retries
- **retry_interval** – delay between retries in seconds
- **kwargs** – other arguments for aiohttp.ClientSession.request

Variables

- **url** – *str*, request URL
- **method** – *str*, request method
- **content_type** – *str*, Content-Type header ("application/x-www-form-urlencoded" by default)
- **timeout** – *float* or *tuple*, request timeout
- **n_retries** – *int*, maximum number of retries
- **success_codes** – *list*-like, list of response codes that indicate request's success
- **retry_interval** – *float*, delay between retries in seconds

process()

Process the response.

Returns depends on *self.process_json()*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

send()

Actually send the request

Returns aiohttp.ClientResponse (*self.response*)

class yadisk_async.api.auth.**RefreshTokenRequest**(*session*, *refresh_token*, *client_id*, *client_secret*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to refresh an existing token.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **refresh_token** – the refresh token that was received with the original token
- **client_id** – application ID
- **client_secret** – application secret password

Returns *TokenObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.auth.**RevokeTokenRequest**(*session*, *token*, *client_id*, *client_secret*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to revoke the token.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **token** – the token to be revoked
- **client_id** – application ID
- **client_secret** – application secret password

Returns *TokenRevokeStatusObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.auth.**GetTokenRequest**(*session*, *code*, *client_id*, *client_secret*, *device_id=None*, *device_name=None*, ***kwargs*)

Bases: yadisk_async.api.api_request.APIRequest

A request to get the token.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **code** – confirmation code
- **client_id** – application ID
- **client_secret** – application secret password
- **device_id** – unique device ID (between 6 and 50 characters)

Returns *TokenObject*

process_json(*js*)

Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.disk.**DiskInfoRequest**(*session*, *fields=None*, ***kwargs*)

Bases: yadisk_async.api.api_request.APIRequest

A request to get disk information.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **fields** – list of keys to be included in the response

Returns *DiskInfoObject*

process_json(*js*)

Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**GetPublicResourcesRequest**(*session*, *offset=0*, *limit=20*, *preview_size=None*, *preview_crop=None*, *type=None*, *fields=None*, ***kwargs*)

Bases: yadisk_async.api.api_request.APIRequest

A request to get a list of public resources.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **offset** – offset from the beginning of the list

- **limit** – maximum number of elements in the list
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **type** – filter based on type of resources ("file" or "dir")
- **fields** – list of keys to be included in the response

Returns *PublicResourcesListObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**UnpublishRequest**(*session*, *path*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to make a public resource private.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource to be unpublished
- **fields** – list of keys to be included in the response

Returns *LinkObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**GetDownloadLinkRequest**(*session*, *path*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to get a download link to a resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource to be downloaded
- **fields** – list of keys to be included in the response

Returns *LinkObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**GetTrashRequest**(*session*, *path=None*, *offset=0*, *limit=20*, *sort=None*, *preview_size=None*, *preview_crop=None*, *fields=None*, ***kwargs*)

Bases: yadisk_async.api.api_request.APIRequest

A request to get meta-information about a trash resource.

Parameters

- **path** – path to the trash resource
- **limit** – number of children resources to be included in the response
- **offset** – number of children resources to be skipped in the response
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **sort** – *str*, field to be used as a key to sort children resources
- **fields** – list of keys to be included in the response

Returns *TrashResourceObject*

process_json(*js*)

Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**RestoreTrashRequest**(*session*, *path*, *dst_path=None*, *force_async=False*, *overwrite=False*, *fields=None*, ***kwargs*)

Bases: yadisk_async.api.api_request.APIRequest

A request to restore trash.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the trash resource to be restored
- **dst_path** – destination path
- **force_async** – forces the operation to be executed asynchronously
- **overwrite** – *bool*, determines whether the destination can be overwritten
- **fields** – list of keys to be included in the response

Returns *LinkObject* or *OperationLinkObject*

process_json(*js*)

Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**DeleteTrashRequest**(*session*, *path=None*, *force_async=False*, *fields=None*, ***kwargs*)

Bases: yadisk_async.api.api_request.APIRequest

A request to delete a trash resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the trash resource to be deleted
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response

Returns *OperationLinkObject* or *None*

process_json(*js*)

Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**LastUploadedRequest**(*session*, *limit=20*, *media_type=None*, *preview_size=None*, *preview_crop=None*, *fields=None*, ***kwargs*)

Bases: yadisk_async.api.api_request.APIRequest

A request to get the list of latest uploaded files sorted by upload date.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **limit** – maximum number of elements in the list
- **media_type** – type of files to include in the list
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **fields** – list of keys to be included in the response

Returns *LastUploadedResourceListObject*

process_json(*js*)

Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**CopyRequest**(*session*, *src_path*, *dst_path*, *overwrite=False*, *force_async=False*, *fields=None*, ***kwargs*)

Bases: yadisk_async.api.api_request.APIRequest

A request to copy a file or a directory.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **src_path** – source path
- **dst_path** – destination path
- **overwrite** – if *True* the destination path can be overwritten, otherwise, an error will be raised
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response

Returns *LinkObject* or *OperationLinkObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**GetMetaRequest**(*session*, *path*, *limit=None*, *offset=None*, *preview_size=None*, *preview_crop=None*, *sort=None*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to get meta-information about a resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource
- **limit** – number of children resources to be included in the response
- **offset** – number of children resources to be skipped in the response
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **sort** – *str*, field to be used as a key to sort children resources
- **fields** – list of keys to be included in the response

Returns *ResourceObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**GetUploadLinkRequest**(*session*, *path*, *overwrite=False*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to get an upload link.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to be uploaded at
- **overwrite** – *bool*, determines whether to overwrite the destination
- **fields** – list of keys to be included in the response

Returns *ResourceUploadLinkObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**MkdirRequest**(*session*, *path*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to create a new directory.

Parameters

- **path** – path to the directory to be created
- **fields** – list of keys to be included in the response

Returns *LinkObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**PublishRequest**(*session*, *path*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to make a resource public.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource to be published
- **fields** – list of keys to be included in the response

Returns *LinkObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**UploadURLRequest**(*session*, *url*, *path*, *disable_redirects=False*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to upload a file from URL.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **url** – source URL
- **path** – destination path
- **disable_redirects** – *bool*, forbid redirects
- **fields** – list of keys to be included in the response

Returns *OperationLinkObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**DeleteRequest**(*session*, *path*, *permanently=False*, *md5=None*, *force_async=False*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to delete a file or a directory.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource to be removed
- **permanently** – if *True*, the resource will be removed permanently, otherwise, it will be just moved to the trash
- **force_async** – forces the operation to be executed asynchronously
- **md5** – *str*, MD5 hash of the file to remove
- **fields** – list of keys to be included in the response

Returns *OperationLinkObject* or *None*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**SaveToDiskRequest**(*session*, *public_key*, *name=None*, *path=None*, *save_path=None*, *force_async=False*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to save a public resource to the disk.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **public_key** – public key or public URL of the resource

- **name** – filename of the saved resource
- **path** – path to the copied resource in the public folder
- **save_path** – path to the destination directory (downloads directory by default)
- **force_async** – forces the operation to be executed asynchronously
- **fields** – list of keys to be included in the response

Returns *LinkObject* or *OperationLinkObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**GetPublicMetaRequest**(*session*, *public_key*, *offset=0*, *limit=20*, *path=None*, *sort=None*, *preview_size=None*, *preview_crop=None*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to get meta-information about a public resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **public_key** – public key or public URL of the resource
- **path** – relative path to a resource in a public folder. By specifying the key of the published folder in *public_key*, you can request metainformation for any resource in the folder.
- **offset** – offset from the beginning of the list of nested resources
- **limit** – maximum number of nested elements to be included in the list
- **sort** – *str*, field to be used as a key to sort children resources
- **preview_size** – file preview size
- **preview_crop** – *bool*, allow preview crop
- **fields** – list of keys to be included in the response

Returns *PublicResourceObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**GetPublicDownloadLinkRequest**(*session*, *public_key*, *path=None*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to get a download link for a public resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **public_key** – public key or public URL of the resource
- **path** – relative path to the resource within the public folder
- **fields** – list of keys to be included in the response

Returns *LinkObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**MoveRequest**(*session*, *src_path*, *dst_path*, *force_async=False*, *overwrite=False*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to move a resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **src_path** – source path to be moved
- **dst_path** – destination path
- **force_async** – forces the operation to be executed asynchronously
- **overwrite** – *bool*, determines whether to overwrite the destination
- **fields** – list of keys to be included in the response

Returns *OperationLinkObject* or *LinkObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**FilesRequest**(*session*, *offset=0*, *limit=20*, *media_type=None*, *preview_size=None*, *preview_crop=None*, *sort=None*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to get a flat list of all files (that doesn't include directories).

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **offset** – offset from the beginning of the list

- **limit** – number of list elements to be included
- **media_type** – type of files to include in the list
- **sort** – *str*, field to be used as a key to sort children resources
- **preview_size** – size of the file preview
- **preview_crop** – *bool*, cut the preview to the size specified in the *preview_size*
- **fields** – list of keys to be included in the response

Returns *FilesResourceListObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.resources.**PatchRequest**(*session*, *path*, *properties*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to update custom properties of a resource.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **path** – path to the resource
- **properties** – *dict*, custom properties to update
- **fields** – list of keys to be included in the response

Returns *ResourceObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

class yadisk_async.api.operations.**GetOperationStatusRequest**(*session*, *operation_id*, *fields=None*, ***kwargs*)
Bases: yadisk_async.api.api_request.APIRequest

A request to get operation status.

Parameters

- **session** – an instance of *yadisk_async.session.SessionWithHeaders* with prepared headers
- **operation_id** – operation ID or link
- **fields** – list of keys to be included in the response

Returns *OperationStatusObject*

process_json(*js*)
Process the JSON response.

Parameters **js** – *dict*, JSON response

Returns processed response, can be anything

CHAPTER 3

Changelog

- **Release 1.3.0 (2019-07-06)**
  - Modified the original library (yadisk) to support *async/await*
  - The library was renamed to *yadisk-async*

The following releases are for yadisk, the original library:

- **Release 1.2.14 (2019-03-26)**
  - Fixed a *TypeError* in *get_public_** functions when passing *path* parameter (see issue #7)
  - Added *unlimited_autoupload_enabled* attribute for *DiskInfoObject*
- **Release 1.2.13 (2019-02-23)**
  - Added *md5* parameter for *remove()*
  - Added *UserPublicInfoObject*
  - Added *country* attribute for *UserObject*
  - Added *photoslice_time* attribute for *ResourceObject*, *PublicResourceObject* and *TrashResourceObject*
- **Release 1.2.12 (2018-10-11)**
  - Fixed *fields* parameter not working properly in *listdir()* (issue #4)
- **Release 1.2.11 (2018-06-30)**
  - Added the missing parameter *sort* for *get_meta()*
  - Added *file* and *antivirus_status* attributes for *ResourceObject*, *PublicResourceObject* and *TrashResourceObject*
  - Added *headers* parameter
  - Fixed a typo in *download()* and *download_public()* (issue #2)
  - Removed **args* parameter everywhere
- **Release 1.2.10 (2018-06-14)**

  - Fixed *timeout=None* behavior. *None* is supposed to mean 'no timeout' but in the older versions it was synonymous with the default timeout.

- **Release 1.2.9 (2018-04-28)**
  - Changed the license to LGPLv3 (see *COPYING* and *COPYING.lesser*)
  - Other package info updates
- **Release 1.2.8 (2018-04-17)**
  - Fixed a couple of typos: *PublicResourceListObject.items* and *TrashResourceListObject.items* had wrong types
  - Substitute field aliases in *fields* parameter when performing API requests (e.g. *embedded -> _embedded*)
- **Release 1.2.7 (2018-04-15)**
  - Fixed a file rewinding bug when uploading/downloading files after a retry
- **Release 1.2.6 (2018-04-13)**
  - Now caching *requests* sessions so that open connections can be reused (which can significantly speed things up sometimes)
  - Disable *keep-alive* when uploading/downloading files by default
- **Release 1.2.5 (2018-03-31)**
  - Fixed an off-by-one bug in *utils.auto_retry()* (which could sometimes result in *AttributeError*)
  - Retry the whole request for *upload()*, *download()* and *download_public()*
  - Set *stream=True* for *download()* and *download_public()*
  - Other minor fixes
- **Release 1.2.4 (2018-02-19)**
  - Fixed *TokenObject* having *exprires_in* instead of *expires_in* (fixed a typo)
- **Release 1.2.3 (2018-01-20)**
  - Fixed a *TypeError* when *WrongResourceTypeError* is raised
- **Release 1.2.2 (2018-01-19)**
  - *refresh_token()* no longer requires a valid or empty token.
- **Release 1.2.1 (2018-01-14)**
  - Fixed auto retries not working. Whoops.
- **Release 1.2.0 (2018-01-14)**
  - Fixed passing *n_retries=0* to *upload()*, *download()* and *download_public()*
  - *upload()*, *download()* and *download_public()* no longer return anything (see the docs)
  - Added *utils* module (see the docs)
  - Added *RetriableYaDiskError*, *WrongResourceTypeError*, *BadGatewayError* and *GatewayTimeoutError*
  - *listdir()* now raises *WrongResourceTypeError* instead of *NotADirectoryError*
- **Release 1.1.1 (2017-12-29)**
  - Fixed argument handling in *upload()*, *download()* and *download_public()*. Previously, passing *n_retries* and *retry_interval* would raise an exception (*TypeError*).

- **Release 1.1.0 (2017-12-27)**
  - Better exceptions (see the docs)
  - Added support for *force_async* parameter
  - Minor bug fixes
- **Release 1.0.8 (2017-11-29)**
  - Fixed yet another *listdir()* bug
- **Release 1.0.7 (2017-11-04)**
  - Added *install_requires* argument to *setup.py*
- **Release 1.0.6 (2017-11-04)**
  - Return *OperationLinkObject* in some functions
- **Release 1.0.5 (2017-10-29)**
  - Fixed *setup.py* to exclude tests
- **Release 1.0.4 (2017-10-23)**
  - Fixed bugs in *upload*, *download* and *listdir* functions
  - Set default *listdir limit* to *10000*
- **Release 1.0.3 (2017-10-22)**
  - Added settings
- **Release 1.0.2 (2017-10-19)**
  - Fixed *get_code_url* function (added missing parameters)
- **Release 1.0.1 (2017-10-18)**
  - Fixed a major bug in *GetTokenRequest* (added missing parameter)
- **Release 1.0.0 (2017-10-18)**
  - Initial release

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

y

Index

I

L

M

N

O

P

R

S

T

U

W

Y