nextcloud-API Documentation

Release 0.0.1

EnterpriseyIntranet

Jul 11, 2020

Contents:

CHAPTER 1

Introduction

Nextcloud-API is Python (2 and 3) wrapper for NextCloud's API. With it you can manage your NextCloud instances from Python scripts.

If you have any question, remark or if you find a bug, don't hesitate to open an issue.

1.1 Quick start

First, create your NextCloud instance:

```
import sys
import os
from os.path import dirname
from os.path import join

sys.path.insert(0, join(dirname(__file__), 'src'))

from nextcloud import NextCloud

NEXTCLOUD_URL = "http://{}:80".format(os.environ['NEXTCLOUD_HOSTNAME'])
NEXTCLOUD_USERNAME = os.environ.get('NEXTCLOUD_ADMIN_USER')
NEXTCLOUD_PASSWORD = os.environ.get('NEXTCLOUD_ADMIN_PASSWORD')

# True if you want to get response as JSON
# False if you want to get response as XML
to_js = True

nxc = NextCloud(endpoint=NEXTCLOUD_URL, user=NEXTCLOUD_USERNAME, password=NEXTCLOUD_
↪PASSWORD, json_output=to_js)
```

Then you can work with NextCloud objects:

```
nxc.get_users()
new_user_id = "new_user_username"
add_user_res = nxc.add_user(new_user_id, "new_user_password321_123")
group_name = "new_group_name"
add_group_res = nxc.add_group(group_name)
add_to_group_res = nxc.add_to_group(new_user_id, group_name)
```

1.2 Which API does it support?

API name	Implementation status	Last checked date
User provisioning API	OK	2019-02-02
OCS Share API	Partially implemented	2019-02-02
WebDAV API	OK	2019-02-02
Activity app API	OK	2019-02-02
Notifications app API	OK	2019-02-02
The LDAP configuration API	OK	2019-02-02
Capabilities API	OK	2019-02-02
Group Folders API	OK	2019-02-02

1.3 Download and install

```
python setup.py install
```

1.4 License

Nextcloud-API is licensed under the GNU General Public License v3.0.

1.5 What's next ?

Check *Examples* and *Nextcloud-API*.

CHAPTER 2

Examples

2.1 Users API methods

```
import sys
import os
from os.path import dirname
from os.path import join

sys.path.insert(0, join(dirname(__file__), 'src'))

from nextcloud import NextCloud

NEXTCLOUD_URL = "http://{}:80".format(os.environ['NEXTCLOUD_HOSTNAME'])
NEXTCLOUD_USERNAME = os.environ.get('NEXTCLOUD_ADMIN_USER')
NEXTCLOUD_PASSWORD = os.environ.get('NEXTCLOUD_ADMIN_PASSWORD')

# True if you want to get response as JSON
# False if you want to get response as XML
to_js = True

nxc = NextCloud(endpoint=NEXTCLOUD_URL, user=NEXTCLOUD_USERNAME, password=NEXTCLOUD_
↪PASSWORD, json_output=to_js)

# Quick start
nxc.get_users()
new_user_id = "new_user_username"
add_user_res = nxc.add_user(new_user_id, "new_user_password321_123")
group_name = "new_group_name"
add_group_res = nxc.add_group(group_name)
add_to_group_res = nxc.add_to_group(new_user_id, group_name)
# End quick start

assert add_group_res.status_code == 100
assert new_user_id in nxc.get_group(group_name).data['users']
```

(continues on next page)

(continued from previous page)

```
assert add_user_res.status_code == 100
assert add_to_group_res.status_code == 100

# remove user
remove_user_res = nxc.delete_user(new_user_id)
assert remove_user_res.status_code == 100
user_res = nxc.get_user(new_user_id)
assert user_res.status_code == 404
```

CHAPTER 3

Nextcloud-API

3.1 nextcloud module

class nextcloud.NextCloud.**NextCloud**(*endpoint*, *user*, *password*, *json_output=True*)
Bases: object

get_connection_issues()
Return Falsy falue if everything is OK, or string representing the connection problem (bad hostname, password, whatever)

class nextcloud.NextCloud.**OCSRequester**(*endpoint*, *user*, *passwd*, *json_output=False*)
Bases: nextcloud.requester.Requester

Requester for OCS API

rtn(*resp*)

nextcloud.NextCloud.**WEBDAV_CLASS**
alias of nextcloud.api_wrappers.webdav.WebDAV

class nextcloud.NextCloud.**WebDAVRequester**(**args*, ***kwargs*)
Bases: nextcloud.requester.Requester

Requester for WebDAV API

copy(*url*, *destination*, *overwrite=False*)

download(*url=''*, *params=None*)

make_collection(*additional_url=''*)

move(*url*, *destination*, *overwrite=False*)

propfind(*additional_url=''*, *headers=None*, *data=None*)

proppatch(*additional_url=''*, *data=None*)

report(*additional_url=''*, *data=None*)

rtn(*resp*, *data=None*)

3.2 NextCloud api wrappers

class nextcloud.api_wrappers.**Activity**(*requester*)
Bases: nextcloud.base.WithRequester

API_URL = '/ocs/v2.php/apps/activity/api/v2/activity'

SUCCESS_CODE = 200

get_activities(*since=None*, *limit=None*, *object_type=None*, *object_id=None*, *sort=None*)
Get an activity feed showing your file changes and other interesting things going on in your Nextcloud

All args are optional

Parameters

- **since** (*int*) – ID of the last activity that you've seen
- **limit** (*int*) – How many activities should be returned (default: 50)
- **object_type** (*string*) – Filter the activities to a given object. May only appear together with object_id
- **object_id** (*string*) – Filter the activities to a given object. May only appear together with object_type
- **sort** (*str, "asc" or "desc"*) – Sort activities ascending or descending (from the since) (Default: desc)

Returns:

class nextcloud.api_wrappers.**Apps**(*requester*)
Bases: nextcloud.base.WithRequester

API_URL = '/ocs/v1.php/cloud/apps'

SUCCESS_CODE = 100

disable_app(*app_id*)
Disable the specified app

Parameters **app_id** – str, app id

Returns

enable_app(*app_id*)
Enable an app

Parameters **app_id** – str, app id

Returns

get_app(*app_id*)
Provide information on a specific application

Parameters **app_id** – str, app id

Returns

get_apps(*filter=None*)
Get a list of apps installed on the Nextcloud server

Parameters **filter** – str, optional "enabled" or "disabled"

Returns

class nextcloud.api_wrappers.**Capabilities**(*requester*)
Bases: nextcloud.base.WithRequester

API_URL = '/ocs/v1.php/cloud/capabilities'

SUCCESS_CODE = 100

get_capabilities()
Obtain capabilities provided by the Nextcloud server and its apps

class nextcloud.api_wrappers.**FederatedCloudShare**(*requester*)
Bases: nextcloud.base.WithRequester

API_URL = '/ocs/v2.php/apps/files_sharing/api/v1'

FEDERATED = 'remote_shares'

SUCCESS_CODE = 200

accept_pending_federated_cloudshare(*sid*)

decline_pending_federated_cloudshare(*sid*)

delete_accepted_federated_cloudshare(*sid*)

get_federated_url(*additional_url=''*)

get_known_federated_cloudshare(*sid*)

list_accepted_federated_cloudshares()

list_pending_federated_cloudshares()

class nextcloud.api_wrappers.**Group**(*requester*)
Bases: nextcloud.base.WithRequester

API_URL = '/ocs/v1.php/cloud/groups'

SUCCESS_CODE = 100

add_group(*gid*)
Add a new group

Parameters **gid** – str, group name

Returns

delete_group(*gid*)
Remove a group

Parameters **gid** – str, group name

Returns

get_group(*gid*)
Retrieve a list of group members

Parameters **gid** – str, group name

Returns

get_groups(*search=None*, *limit=None*, *offset=None*)
Retrieve a list of groups from the Nextcloud server

Parameters

- **search** – string, optional search string
- **limit** – int, optional limit value

- **offset** – int, optional offset value

Returns

get_subadmins(*gid*)
List subadmins of the group

Parameters **gid** – str, group name

Returns

class nextcloud.api_wrappers.**GroupFolders**(*requester*)
Bases: nextcloud.base.WithRequester

API_URL = '/apps/groupfolders/folders'

SUCCESS_CODE = 100

create_group_folder(*mountpoint*)
Create a new group folder

Parameters **mountpoint** (*str*) – name for the new folder

Returns:

delete_group_folder(*fid*)
Delete a group folder

Parameters **fid** (*int/str*) – group folder id

Returns:

get_group_folder(*fid*)
Return a specific configured folder and it's settings

Parameters **fid** (*int/str*) – group folder id

Returns:

get_group_folders()
Return a list of call configured folders and their settings

Returns:

grant_access_to_group_folder(*fid*, *gid*)
Give a group access to a folder

Parameters

- **fid** (*int/str*) – group folder id
- **gid** (*str*) – group to share with id

Returns:

rename_group_folder(*fid*, *mountpoint*)
Change the name of a folder

Parameters

- **fid** (*int/str*) – group folder id
- **mountpoint** (*str*) – The new name for the folder

Returns:

revoke_access_to_group_folder(*fid*, *gid*)
Remove access from a group to a folder

Parameters

- **fid** (*int/str*) – group folder id
- **gid** (*str*) – group id

Returns:

set_permissions_to_group_folder(*fid*, *gid*, *permissions*)
Set the permissions a group has in a folder

Parameters

- **fid** (*int/str*) – group folder id
- **gid** (*str*) – group id
- **permissions** (*int*) – The new permissions for the group as attribute of Permission class

Returns:

set_quota_of_group_folder(*fid*, *quota*)
Set the quota for a folder in bytes

Parameters

- **fid** (*int/str*) – group folder id
- **quota** (*int/str*) – The new quota for the folder in bytes, user -3 for unlimited

Returns:

class nextcloud.api_wrappers.**Notifications**(*requester*)
Bases: nextcloud.base.WithRequester

API_URL = '/ocs/v2.php/apps/notifications/api/v2/notifications'

SUCCESS_CODE = 200

delete_all_notifications()
Delete all notification for a logged in user

Notes

This endpoint was added for Nextcloud 14

delete_notification(*notification_id*)
Delete single notification by id for a user

Parameters **notification_id** (*int*) – Notification id

Returns:

get_notification(*notification_id*)
Get single notification by id for a user

Parameters **notification_id** (*int*) – Notification id

Returns:

get_notifications()
Get list of notifications for a logged in user

class nextcloud.api_wrappers.**Share**(*requester*)
Bases: nextcloud.base.WithRequester

API_URL = '/ocs/v2.php/apps/files_sharing/api/v1'

LOCAL = 'shares'

SUCCESS_CODE = 200

create_share(*path*, *share_type*, *share_with=None*, *public_upload=None*, *password=None*, *permissions=None*)

Share a file/folder with a user/group or as public link

Mandatory fields: share_type, path and share_with for share_type USER (0) or GROUP (1).

Parameters

- **path** (*str*) – path to the file/folder which should be shared
- **share_type** (*int*) – ShareType attribute
- **share_with** (*str*) – user/group id with which the file should be shared
- **public_upload** (*bool*) – bool, allow public upload to a public shared folder (true/false)
- **password** (*str*) – password to protect public link Share with
- **permissions** (*int*) – sum of selected Permission attributes

Returns:

delete_share(*sid*)

Remove the given share

Parameters **sid** (*str*) – share id

Returns:

get_local_url(*additional_url=''*)

get_share_info(*sid*)

Get information about a given share

Parameters **sid** (*int*) – share id

Returns:

get_shares()

Get all shares from the user

get_shares_from_path(*path*, *reshares=None*, *subfiles=None*)

Get all shares from a given file/folder

Parameters

- **path** (*str*) – path to file/folder
- **reshares** (*bool*) – (optional) return not only the shares from the current user but all shares from the given file
- **subfiles** (*bool*) – (optional) return all shares within a folder, given that path defines a folder

Returns:

update_share(*sid*, *permissions=None*, *password=None*, *public_upload=None*, *expire_date=''*)

Update a given share, only one value can be updated per request

Parameters

- **sid** (*str*) – share id

- **permissions** (*int*) – sum of selected Permission attributes
- **password** (*str*) – password to protect public link Share with
- **public_upload** (*bool*) – bool, allow public upload to a public shared folder (true/false)
- **expire_date** (*str*) – set an expire date for public link shares. Format: 'YYYY-MM-DD'

Returns:

static validate_share_parameters(*path*, *share_type*, *share_with*)
Check if share parameters make sense

Parameters

- **path** (*str*) – path to the file/folder which should be shared
- **share_type** (*int*) – ShareType attribute
- **share_with** (*str*) – user/group id with which the file should be shared

Returns True if parameters make sense together, False otherwise

Return type bool

class nextcloud.api_wrappers.**User**(*requester*)
Bases: nextcloud.base.WithRequester

API_URL = '/ocs/v1.php/cloud/users'

SUCCESS_CODE = 100

add_to_group(*uid*, *gid*)
Add the specified user to the specified group

Parameters

- **uid** – str, uid of user
- **gid** – str, name of group

Returns

add_user(*uid*, *passwd*)
Create a new user on the Nextcloud server

Parameters

- **uid** – str, uid of new user
- **passwd** – str, password of new user

Returns

create_subadmin(*uid*, *gid*)
Make a user the subadmin of a group

Parameters

- **uid** – str, uid of user
- **gid** – str, name of group

Returns

delete_user(*uid*)
Delete a user from the Nextcloud server

Parameters **uid** – str, uid of user

Returns

disable_user(*uid*)
Disable a user on the Nextcloud server so that the user cannot login anymore

Parameters **uid** – str, uid of user

Returns

edit_user(*uid*, *what*, *value*)
Edit attributes related to a user

Users are able to edit email, displayname and password; admins can also edit the quota value

Parameters

- **uid** – str, uid of user
- **what** – str, the field to edit
- **value** – str, the new value for the field

Returns

enable_user(*uid*)
Enable a user on the Nextcloud server so that the user can login again

Parameters **uid** – str, uid of user

Returns

get_subadmin_groups(*uid*)
Get the groups in which the user is a subadmin

Parameters **uid** – str, uid of user

Returns

get_user(*uid*)
Retrieve information about a single user

Parameters **uid** – str, uid of user

Returns

get_users(*search=None*, *limit=None*, *offset=None*)
Retrieve a list of users from the Nextcloud server

Parameters

- **search** – string, optional search string
- **limit** – int, optional limit value
- **offset** – int, optional offset value

Returns

remove_from_group(*uid*, *gid*)
Remove the specified user from the specified group

Parameters

- **uid** – str, uid of user
- **gid** – str, name of group

Returns

remove_subadmin(*uid*, *gid*)
Remove the subadmin rights for the user specified from the group specified

Parameters

- **uid** – str, uid of user
- **gid** – str, name of group

Returns

resend_welcome_mail(*uid*)
Trigger the welcome email for this user again

Parameters **uid** – str, uid of user

Returns

class nextcloud.api_wrappers.**UserLDAP**(*requester*)
Bases: nextcloud.base.WithRequester

API_URL = '/ocs/v2.php/apps/user_ldap/api/v1/config'

CONFIG_KEYS = ['ldapHost', 'ldapPort', 'ldapBackupHost', 'ldapBackupPort', 'ldapBase',

SUCCESS_CODE = 200

create_ldap_config()
Create a new and empty LDAP configuration

delete_ldap_config(*config_id*)
Delete a given LDAP configuration

Parameters **config_id** (*str*) – User LDAP config id

Returns:

edit_ldap_config(*config_id*, *data*)
Update a configuration with the provided values

You can find list of all config keys in get_ldap_config method response or in Nextcloud docs

Parameters

- **config_id** (*str*) – User LDAP config id
- **data** (*dict*) – config values to update

Returns:

get_ldap_agent_name(*config_id*)

get_ldap_agent_password(*config_id*)

get_ldap_attributes_for_group_search(*config_id*)

get_ldap_attributes_for_user_search(*config_id*)

get_ldap_backup_host(*config_id*)

get_ldap_backup_port(*config_id*)

get_ldap_base(*config_id*)

get_ldap_base_groups(*config_id*)

get_ldap_base_users(*config_id*)

get_ldap_cache_ttl(*config_id*)

get_ldap_config(*config_id*, *show_password=None*)
Get all keys and values of the specified LDAP configuration

Parameters

- **config_id** (*str*) – User LDAP config id
- **show_password** (*int*) – 0 or 1 whether to return the password in clear text (default 0)

Returns:

get_ldap_config_id(*idx=1*)
The LDAP config ID is a string. Given the number of the config file, return the corresponding string ID if the configuration exists.

Parameters **idx** – The index of the configuration. If a single configuration exists on the server from the beginning, it is going to have index of 1.

Returns Configuration string or None

get_ldap_configuration_active(*config_id*)

get_ldap_default_ppolicy_dn(*config_id*)

get_ldap_dynamic_group_member_url(*config_id*)

get_ldap_email_attribute(*config_id*)

get_ldap_experienced_admin(*config_id*)

get_ldap_expert_username_attr(*config_id*)

get_ldap_expert_uuidgroup_attr(*config_id*)

get_ldap_expert_uuiduser_attr(*config_id*)

get_ldap_gid_number(*config_id*)

get_ldap_group_display_name(*config_id*)

get_ldap_group_filter(*config_id*)

get_ldap_group_filter_groups(*config_id*)

get_ldap_group_filter_mode(*config_id*)

get_ldap_group_filter_objectclass(*config_id*)

get_ldap_group_member_assoc_attr(*config_id*)

get_ldap_has_member_of_filter_support(*config_id*)

get_ldap_home_folder_naming_rule(*config_id*)

get_ldap_host(*config_id*)

get_ldap_last_jpeg_photo_lookup(*config_id*)

get_ldap_login_filter(*config_id*)

get_ldap_login_filter_attributes(*config_id*)

get_ldap_login_filter_email(*config_id*)

get_ldap_login_filter_mode(*config_id*)

get_ldap_login_filter_username(*config_id*)

get_ldap_lowest_existing_config_id(*lower_bound=1*, *upper_bound=10*)
Given (inclusive) lower and upper bounds, try to guess an existing LDAP config ID that corresponds to an index within those bounds.

Parameters

- **lower_bound** – The lowest index of the configuration possible.
- **upper_bound** – The greatest index of the configuration possible.

Returns Configuration string or None

get_ldap_nested_groups(*config_id*)

get_ldap_override_main_server(*config_id*)

get_ldap_paging_size(*config_id*)

get_ldap_port(*config_id*)

get_ldap_quota_attribute(*config_id*)

get_ldap_quota_default(*config_id*)

get_ldap_tls(*config_id*)

get_ldap_turn_off_cert_check(*config_id*)

get_ldap_turn_on_password_change(*config_id*)

get_ldap_use_member_of_to_detect_membership(*config_id*)

get_ldap_user_display_name(*config_id*)

get_ldap_user_filter(*config_id*)

get_ldap_user_filter_groups(*config_id*)

get_ldap_user_filter_mode(*config_id*)

get_ldap_user_filter_objectclass(*config_id*)

get_ldap_uuid_group_attribute(*config_id*)

get_ldap_uuid_user_attribute(*config_id*)

ldap_cache_flush(*config_id*)
Flush the cache, so the fresh LDAP DB data is used.

Implementation detail: This is performed by a fake update of LDAP cache TTL as indicated by

Parameters **config_id** (*str*) – User LDAP config id

set_ldap_agent_name(*config_id*, *value*)

set_ldap_agent_password(*config_id*, *value*)

set_ldap_attributes_for_group_search(*config_id*, *value*)

set_ldap_attributes_for_user_search(*config_id*, *value*)

set_ldap_backup_host(*config_id*, *value*)

set_ldap_backup_port(*config_id*, *value*)

set_ldap_base(*config_id*, *value*)

set_ldap_base_groups(*config_id*, *value*)

set_ldap_base_users(*config_id*, *value*)

set_ldap_cache_ttl(*config_id*, *value*)

set_ldap_configuration_active(*config_id*, *value*)

set_ldap_default_ppolicy_dn(*config_id*, *value*)

set_ldap_dynamic_group_member_url(*config_id*, *value*)

set_ldap_email_attribute(*config_id*, *value*)

set_ldap_experienced_admin(*config_id*, *value*)

set_ldap_expert_username_attr(*config_id*, *value*)

set_ldap_expert_uuidgroup_attr(*config_id*, *value*)

set_ldap_expert_uuiduser_attr(*config_id*, *value*)

set_ldap_gid_number(*config_id*, *value*)

set_ldap_group_display_name(*config_id*, *value*)

set_ldap_group_filter(*config_id*, *value*)

set_ldap_group_filter_groups(*config_id*, *value*)

set_ldap_group_filter_mode(*config_id*, *value*)

set_ldap_group_filter_objectclass(*config_id*, *value*)

set_ldap_group_member_assoc_attr(*config_id*, *value*)

set_ldap_has_member_of_filter_support(*config_id*, *value*)

set_ldap_home_folder_naming_rule(*config_id*, *value*)

set_ldap_host(*config_id*, *value*)

set_ldap_last_jpeg_photo_lookup(*config_id*, *value*)

set_ldap_login_filter(*config_id*, *value*)

set_ldap_login_filter_attributes(*config_id*, *value*)

set_ldap_login_filter_email(*config_id*, *value*)

set_ldap_login_filter_mode(*config_id*, *value*)

set_ldap_login_filter_username(*config_id*, *value*)

set_ldap_nested_groups(*config_id*, *value*)

set_ldap_override_main_server(*config_id*, *value*)

set_ldap_paging_size(*config_id*, *value*)

set_ldap_port(*config_id*, *value*)

set_ldap_quota_attribute(*config_id*, *value*)

set_ldap_quota_default(*config_id*, *value*)

set_ldap_tls(*config_id*, *value*)

set_ldap_turn_off_cert_check(*config_id*, *value*)

set_ldap_turn_on_password_change(*config_id*, *value*)

set_ldap_use_member_of_to_detect_membership(*config_id*, *value*)

set_ldap_user_display_name(*config_id*, *value*)

set_ldap_user_filter(*config_id*, *value*)

set_ldap_user_filter_groups(*config_id*, *value*)

set_ldap_user_filter_mode(*config_id*, *value*)

set_ldap_user_filter_objectclass(*config_id*, *value*)

set_ldap_uuid_group_attribute(*config_id*, *value*)

set_ldap_uuid_user_attribute(*config_id*, *value*)

nextcloud.api_wrappers.**WEBDAV_CLASS**
alias of nextcloud.api_wrappers.webdav.WebDAV

class nextcloud.api_wrappers.**WebDAV**(**args*, ***kwargs*)
Bases: nextcloud.base.WithRequester

API_URL = '/remote.php/dav/files'

assure_folder_exists(*uid*, *folder_path*)
Create folder on Nextcloud storage, don't do anything if the folder already exists. :param uid: uid of user :type uid: str :param folder_path: folder path :type folder_path: str

Returns:

assure_tree_exists(*uid*, *tree_path*)
Make sure that the folder structure on Nextcloud storage exists :param uid: uid of user :type uid: str :param folder_path: The folder tree :type folder_path: str

Returns:

copy_path(*uid*, *path*, *destination_path*, *overwrite=False*)
Copy file or folder to destination

Parameters

- **uid** (*str*) – uid of user
- **path** (*str*) – file or folder path to copy
- **destionation_path** (*str*) – destination where to copy
- **overwrite** (*bool*) – allow destination path overriding

create_folder(*uid*, *folder_path*)
Create folder on Nextcloud storage

Parameters

- **uid** (*str*) – uid of user
- **folder_path** (*str*) – folder path

delete_path(*uid*, *path*)
Delete file or folder with all content of given user by path

Parameters

- **uid** (*str*) – uid of user
- **path** (*str*) – file or folder path to delete

download_file(*uid*, *path*)
Download file of given user by path File will be saved to working directory path argument must be valid file path Modified time of saved file will be synced with the file properties in Nextcloud

Exception will be raised if:

- path doesn't exist,
- path is a directory, or if
- file with same name already exists in working directory

Parameters

- **uid** (*str*) – uid of user
- **path** (*str*) – file path

Returns None

list_favorites(*uid*, *path=''*)
Set files of a user favorite

Parameters

- **uid** (*str*) – uid of user
- **path** (*str*) – file or folder path to make favorite

list_folders(*uid*, *path=None*, *depth=1*, *all_properties=False*)
Get path files list with files properties for given user, with given depth

Parameters

- **uid** (*str*) – uid of user
- **path** (*str/None*) – files path
- **depth** (*int*) – depth of listing files (directories content for example)
- **all_properties** (*bool*) – list all available file properties in Nextcloud

Returns list of dicts if json_output list of File objects if not json_output

move_path(*uid*, *path*, *destination_path*, *overwrite=False*)
Move file or folder to destination

Parameters

- **uid** (*str*) – uid of user
- **path** (*str*) – file or folder path to move
- **destionation_path** (*str*) – destination where to move
- **overwrite** (*bool*) – allow destination path overriding

set_favorites(*uid*, *path*)
Set files of a user favorite

Parameters

- **uid** (*str*) – uid of user
- **path** (*str*) – file or folder path to make favorite

upload_file(*uid*, *local_filepath*, *remote_filepath*, *timestamp=None*)
Upload file to Nextcloud storage

Parameters

- **uid** (*str*) – uid of user
- **local_filepath** (*str*) – path to file on local storage

- **remote_filepath** (*str*) – path where to upload file on Nextcloud storage
- **timestamp** (*int*) – timestamp of upload file. If None, get time by local file.

upload_file_contents(*uid*, *file_contents*, *remote_filepath*, *timestamp=None*)
Upload file to Nextcloud storage

Parameters

- **uid** (*str*) – uid of user
- **file_contents** (*bytes*) – Bytes the file to be uploaded consists of
- **remote_filepath** (*str*) – path where to upload file on Nextcloud storage
- **timestamp** (*int*) – mtime of upload file

Python Module Index

n

Index

E

F

G

L

M

N

O

P

R

S

U

V

W