Data Delivery System

SciLifeLab Data Centre

Dec 06, 2021

CONTENTS

Check out the *Installation* section for further information.

CHAPTER

ONE

USAGE

1.1 Installation

When the DDS is in production, you **will be** able to install it using pip:

```
$ pip install dds
```

1.2 Commands

1.2.1 dds

SciLifeLab Data Delivery System (DDS) command line interface.

Access token is saved in a .dds_cli_token file in the home directory.

```
dds [OPTIONS] COMMAND [ARGS]...
```

Options

-v, --verbose
Print verbose output to the console.

-l, --log-file <filename>
Save a log to a file.

--no-prompt
Run without any interactive features.

--version
Show the version and exit.

add-user

Add user to DDS, sending an invitation email to that person.

```
dds add-user [OPTIONS]
```

Options

-u, --username <username>
Your Data Delivery System username.

-e, --email <email>
Required Email of the user you would like to invite.

-r, --role <role>
Required Type of account.

Options Super Admin | Unit Admin | Unit Personnel | Project Owner | Researcher

-p, --project <project>
Existing Project you want the user to be associated to.

auth

Manage the saved authentication token.

```
dds auth [OPTIONS] COMMAND [ARGS]...
```

info

Print info on saved authentication token validity and age.

```
dds auth info [OPTIONS]
```

login

Renew the authentication token stored in the '.dds_cli_token' file.

Run this command before running the cli in a non interactive fashion as this enables the longest possible session time before a password needs to be entered again.

```
dds auth login [OPTIONS]
```

Options

-u, --username <username>
Your Data Delivery System username. Required unless the –*check* flag is used.

logout

Remove the saved authentication token by deleting the '.dds_cli_token' file.

```
dds auth logout [OPTIONS]
```

create

Create a project.

```
dds create [OPTIONS]
```

Options

-u, --username <username>
Your Data Delivery System username.

-t, --title <title>
Required The title of the project

-d, --description <description>
Required A description of the project

-pi, --principal-investigator <principal_investigator>
Required The name of the Principal Investigator

--is_sensitive
Indicate if the Project includes sensitive data

--owner <owner>
Email of a user to be added to the project as Project Owner

--researcher <researcher>
Email of a user to be added to the project as Researcher

get

Download specified files from the cloud and restores the original format.

```
dds get [OPTIONS]
```

Options

-u, --username <username>

Your Data Delivery System username.

-p, --project <project>

Required Project ID from which you're downloading data.

-a, --get-all

Download all project contents.

Default False

-s, --source <source>

Path to file or directory.

-spf, --source-path-file <source_path_file>

File containing path to files or directories.

-d, --destination <destination>

Destination of downloaded files.

--break-on-fail

Cancel download of all files if one fails

Default False

-nt, --num-threads <num_threads>

Number of parallel threads to perform the download.

Default 4

--silent

Turn off progress bar for each individual file. Summary bars still visible.

Default False

--verify-checksum

Perform SHA-256 checksum verification after download (slower).

Default False

ls

List your projects and project files.

To list all projects, run *dds ls* without any arguments.

Specify a Project ID to list the files within a project. You can also follow this with a subfolder path to show files within that folder.

```
dds ls [OPTIONS] [PROJECT ID] [FOLDER]
```

Options

-lp, --projects
List all project connected to your account.

-s, --size
Show size of project contents.

-u, --username <username>
Your Data Delivery System username.

--usage
Show the usage for available projects, in GBHours and cost. No effect when specifying a project id.

Default False

--sort <sort>
Which column to sort the project list by.

Options id | title | pi | status | updated | size | usage | cost

-t, --tree
Display the entire project(s) directory tree

--users
Display users associated with a project(Requires a project id)

Arguments

[PROJECT ID]
Optional argument

FOLDER
Optional argument

put

Process and upload specified files to the cloud.

```
dds put [OPTIONS]
```

Options

-u, --username <username>
Your Data Delivery System username

-p, --project <project>
Required Project ID to which you're uploading data

-s, --source <source>
Path to file or directory (local)

-spf, --source-path-file <source_path_file>
File containing path to files or directories

--break-on-fail
Cancel upload of all files if one fails

Default False

--overwrite
Overwrite files if already uploaded

Default False

-nt, --num-threads <num_threads>
Number of parallel threads to perform the delivery

Default 4

--silent
Turn off progress bar for each individual file. Summary bars still visible. Suggested for uploads including a large number of files.

Default False

rm

Delete the files within a project.

```
dds rm [OPTIONS] [PROJ_ARG]
```

Options

-p, --project <project>
Required Project ID.

-u, --username <username>
Your Data Delivery System username.

-a, --rm-all
Remove all project contents.

-f, --file <file>
Path to file to remove.

-fl, --folder <folder>
Path to folder to remove.

Arguments

PROJ_ARG
Optional argument

1.3 All modules

1.3.1 dds_cli package

Submodules

dds_cli.account_adder

Admin class, adds .

class dds_cli.account_adder.**AccountAdder**(*username: str*, *method: str = 'add'*, *no_prompt: bool = False*)

Bases: *dds_cli.base.DDSBaseClass*

Admin class for adding users, etc.

add_user(*email*, *role*, *project*)

Invite user.

dds_cli.auth

Data Delivery System saved authentication token manager.

class dds_cli.auth.**Auth**(*username: str*, *authenticate: bool = True*)

Bases: *dds_cli.base.DDSBaseClass*

Authentication manager class.

check()

logout()

dds_cli.base

Base class for the DDS CLI. Verifies the users access to the DDS.

class dds_cli.base.**DDSBaseClass**(*username*, *project=None*, *dds_directory: Optional[pathlib.Path] = None*, *method: Optional[str] = None*, *authenticate: bool = True*, *method_check: bool = True*, *force_renew_token: bool = False*, *no_prompt: bool = False*)

Bases: object

Data Delivery System base class. For common operations.

dds_cli.cli_decorators

Module for all decorators related to the execution of the DDS CLI.

dds_cli.cli_decorators.**removal_spinner**(*func*)

Spinner for the rm command

dds_cli.cli_decorators.**subpath_required**(*func*)

Make sure that the subpath to the temporary file directory exist.

dds_cli.cli_decorators.**update_status**(*func*)

Decorator for updating the status of files.

dds_cli.cli_decorators.**verify_proceed**(*func*)

Decorator for verifying that the file is not cancelled. Also cancels the upload of all non-started files if break-on-fail.

dds_cli.data_getter

Data getter.

class dds_cli.data_getter.**DataGetter**(*username: str*, *project: Optional[str] = None*, *break_on_fail: bool = False*, *get_all: bool = False*, *source: tuple = ()*, *source_path_file: Optional[pathlib.Path] = None*, *destination: pathlib.Path = PosixPath('.')*, *silent: bool = False*, *verify_checksum: bool = False*, *method: str = 'get'*, *no_prompt: bool = False*)

Bases: `dds_cli.base.DDSBaseClass`

Data getter class.

download_and_verify(*file*, *progress*)

Download the file, reveals the original data and verifies the integrity.

get(*file*, *progress*, *task*)

Download files from the cloud.

update_db(*file*)

Update file info in db.

dds_cli.data_lister

Data Lister – Lists the projects and project content.

class dds_cli.data_lister.**DataLister**(*username: str*, *method: str = 'ls'*, *project: Optional[str] = None*, *show_usage: bool = False*, *tree: bool = False*, *no_prompt: bool = False*)

Bases: `dds_cli.base.DDSBaseClass`

Data lister class.

format_columns(*total_size=None*, *usage_info=None*)

Define the formatting for the project table according to what is returned from API.

list_files(*folder: Optional[str] = None*, *show_size: bool = False*)

Create a tree displaying the files within the project.

list_projects(*sort_by='Updated'*)

Get a list of project(s) the user is involved in.

list_recursive(*show_size: bool = False*)

Recursively list project contents.

list_users()

Get a list of user(s) involved in a project.

sort_projects(*projects*, *sort_by='id'*)

Sort the projects according to ID and either default or chosen column.

dds_cli.data_putter

Data putter.

class dds_cli.data_putter.**DataPutter**(*username: str*, *project: Optional[str] = None*, *break_on_fail: bool = False*, *overwrite: bool = False*, *source: tuple = ()*, *source_path_file: Optional[pathlib.Path] = None*, *silent: bool = False*, *method: str = 'put'*, *no_prompt: bool = False*)

Bases: *dds_cli.base.DDSBaseClass*

Data putter class.

add_file_db(*file*)
Make API request to add file to DB.

protect_and_upload(*file*, *progress*)
Process and upload the file while handling the progress bars.

put(*file*, *progress*, *task*)
Upload files to the cloud.

dds_cli.data_putter.**put**(*username*, *project*, *source*, *source_path_file*, *break_on_fail*, *overwrite*, *num_threads*, *silent*, *no_prompt*)
Handle upload of data.

dds_cli.data_remover

Data Remover – Removes files from projects.

class dds_cli.data_remover.**DataRemover**(*project: str*, *username: str*, *method: str = 'rm'*, *no_prompt: bool = False*)

Bases: *dds_cli.base.DDSBaseClass*

Data remover class.

static delete_tempfile(*file: pathlib.Path*)
Delete the specified file.

remove_all(**_*, ***__*)
Remove all files in project.

remove_file(*files*)
Remove specific files.

remove_folder(*folder*)
Remove specific folders.

dds_cli.directory

Directory module. Creates the DDS directory during delivery.

class dds_cli.directory.**DDSDirectory**(*path=<class 'pathlib.Path'>*, *add_file_dir: bool = True*)

Bases: object

Data Delivery System directory class.

dds_cli.exceptions

Custom Exception classes.

exception `dds_cli.exceptions.`**`APIError`**

Bases: `Exception`

Error connecting to the dds web server.

exception `dds_cli.exceptions.`**`ApiRequestError`**(*message*)

Bases: `requests.exceptions.RequestException`

Request to REST API failed.

exception `dds_cli.exceptions.`**`ApiResponseError`**(*message*)

Bases: `Exception`

REST API Request does not return code 200 in response.

exception `dds_cli.exceptions.`**`AuthenticationError`**(*message*, *sign=':no_entry_sign:'*)

Bases: `click.exceptions.ClickException`

Errors due to user authentication.

exception `dds_cli.exceptions.`**`DDSCLIException`**(*message*, *sign=':warning-emoji:'*)

Bases: `click.exceptions.ClickException`

Base exception for click in DDS.

exception `dds_cli.exceptions.`**`DownloadError`**

Bases: `Exception`

Errors relating to file download.

exception `dds_cli.exceptions.`**`InvalidMethodError`**(*attempted_method*, *message='Attempting an invalid method in the DDS'*)

Bases: `Exception`

Valid methods are only ls, put, get, rm. Anything else should raise errors.

exception `dds_cli.exceptions.`**`NoDataError`**

Bases: `Exception`

Errors when there is no data to do anything with.

exception `dds_cli.exceptions.`**`S3KeyLengthExceeded`**(*filename*, *message="Exceeds allowed limit for combined path and name length and can't be stored in the system."*)

Bases: *`dds_cli.exceptions.UploadError`*

The name in bucket exceeded the allowed 1024 bytes.

exception `dds_cli.exceptions.`**`TokenNotFoundError`**(*message*, *sign=':warning-emoji:'*)

Bases: *`dds_cli.exceptions.AuthenticationError`*

No token retrieved from REST API or from File.

exception `dds_cli.exceptions.`**`UploadError`**

Bases: `Exception`

Errors relating to file uploads.

dds_cli.file_compressor

“Compressor module. Handles the compression of files.

class dds_cli.file_compressor.**CompressionMagic**

Bases: object

Compression format signatures

BZIP2 = b'BZh'

GZIP = b'\x1f\x8b'

LZIP = b'LZIP'

RAR4 = b'Rar!\x1a\x07\x00'

RAR5 = b'Rar!\x1a\x07\x01\x00'

ZSTANDARD = b'(\xb5/\xfd'

class dds_cli.file_compressor.**Compressor**(*algorithm: str = 'zstandard'*)

Bases: object

Handles operations relating to file compression.

algorithm: str = 'zstandard'

static compress_file(*file: pathlib.Path*, *chunk_size: int = 65536*) → bytes

Compresses file by reading it chunk by chunk.

static decompress_filechunks(*chunks*, *outfile: pathlib.Path*, ***_*)

Decompress file chunks

fmt_magic: dict

is_compressed(*file*)

Checks if a file is compressed or not.

max_magic_len: int

dds_cli.file_encryptor

File encryptor module. Handles the encryption of the files and the generation of the keys.

class dds_cli.file_encryptor.**Decryptor**(*project_keys: tuple*, *peer_public: str*, *key_salt: str*)

Bases: *dds_cli.file_encryptor.ECDHKeyHandler*

Handles the decryption of the files.

decrypt_file(*infile: pathlib.Path*)

Decrypts the file

class dds_cli.file_encryptor.**ECDHKeyHandler**

Bases: object

Generates the shared encryption/decryption key, and transforms the components.

static generate_shared_key(*my_private*, *peer_public*, *salt: bytes = b''*)

Derive the shared key for file encryption.

static get_public_component_hex(*private_key*)

Gets the public key and converts to hex string.

static public_to_hex(*public_key*)
Converts public key to hexstring.

class dds_cli.file_encryptor.**Encryptor**(*project_keys*)
Bases: *dds_cli.file_encryptor.ECDHKeyHandler*

Handles the encryption of the files.

encrypt_filechunks(*chunks*, *outfile: pathlib.Path*, *progress: Optional[tuple] = None*)
Encrypts the file in chunks.

Encrypts the file in chunks using the IETF ratified ChaCha20-Poly1305 construction described in RFC8439 (obsoletes 7539).

static verify_checksum(*file: pathlib.Path*, *correct_checksum*)
Generate file checksum and verify the integrity

dds_cli.file_handler

File handler module. Base class for LocalFileHandler and RemoteFileHandler.

class dds_cli.file_handler.**FileHandler**(*user_input*, *local_destination*, *project=None*)
Bases: object

Main file handler.

static append_errors_to_file(*file: pathlib.Path*, *info*)
Save errors to specific json file.

static create_summary_table(*all_failed_data*, *get_single_files: bool = True*, *upload: bool = True*)
Create summary of files and errors.

static delete_tempdir(*directory: pathlib.Path*)
Delete the specified directory.

dds_cli.file_handler_local

Local file handler module

class dds_cli.file_handler_local.**LocalFileHandler**(*user_input*, *temporary_destination*, *project*)
Bases: *dds_cli.file_handler.FileHandler*

Collects the files specified by the user.

check_previous_upload(*token*)
Do API call and check for the files in the DB.

create_encrypted_name(*raw_file: pathlib.Path*, *subpath: str = ''*, *no_compression: bool = True*)
Create new file name to save encrypted file.

create_upload_status_dict(*existing_files*, *overwrite=False*)
Create dict for tracking file delivery status

static generate_bucket_filepath(*filename=''*, *folder=PosixPath('.')*)
Generates filename and new path which the file will be called in the bucket.

static read_file(*file*, *chunk_size: int = 65536*)
Read file in chunk_size sized chunks.

stream_from_file (*file*)
Read raw or compress file depending on if compressed already or not.

dds_cli.file_handler_remote

Remote file handler module.

class dds_cli.file_handler_remote.**RemoteFileHandler** (*get_all*, *user_input*, *token*, *project*, *destination=PosixPath('.')*)

Bases: *dds_cli.file_handler.FileHandler*

Collects the files specified by the user.

create_download_status_dict ()
Create dict for tracking file download status.

static write_file (*chunks*, *outfile: pathlib.Path*, ***_*)
Write file chunks to file.

dds_cli.project_creator

Data Delivery System Project Creator.

class dds_cli.project_creator.**ProjectCreator** (*username: str*, *method: str = 'create'*, *no_prompt: bool = False*)

Bases: *dds_cli.base.DDSBaseClass*

Project creator class.

create_project (*title*, *description*, *principal_investigator*, *sensitive*, *users_to_add*)
Create project with title and description.

dds_cli.s3_connector

S3 Connector module.

class dds_cli.s3_connector.**S3Connector** (*project_id: dataclasses.InitVar[str]*, *token: dataclasses.InitVar[dict]*)

Bases: object

Connect to Simple Storage Service.

bucketname: str

connect ()
Connect to S3 resource.

keys: dict

project_id: dataclasses.InitVar[str]

resource = None

safespring_project: str

token: dataclasses.InitVar[dict]

url: str

dds_cli.status

Status module.

class dds_cli.status.**DeliveryStatus**

Bases: object

Delivery status class.

UPLOAD_STATUS = {'database': {'finished': False, 'in_progress': False}, 'processing

classmethod cancel_all()

Cancel upload all files.

classmethod cancel_one()

Cancel the failed file.

class dds_cli.status.**ProgressPercentage**(*progress*, *task*)

Bases: object

Updates the progress bar with the callback from boto3.

dds_cli.text_handler

Handles formatting of string.

class dds_cli.text_handler.**StringFormat**

Bases: object

Defines different formats for strings, e.g. colors and bold.

BLUE = '\x1b[94m'

BOLD = '\x1b[1m'

CYAN = '\x1b[96m'

DARKCYAN = '\x1b[36m'

END = '\x1b[0m'

GREEN = '\x1b[92m'

HEADER = '\x1b[95m'

PURPLE = '\x1b[95m'

RED = '\x1b[91m'

UNDERLINE = '\x1b[4m'

YELLOW = '\x1b[93m'

class dds_cli.text_handler.**TextHandler**

Bases: *dds_cli.text_handler.StringFormat*

Handler for text formatting.

static format_tabs(*string_len*, *max_string_len*, *tab_len=4*)

Format number of tabs to have within string.

static task_name(*file*, *step=''*, *max_len=30*)

Generate display name for progress task

dds_cli.timestamp

class dds_cli.timestamp.**TimeStamp**

Bases: object

dds_cli.user

User module.

class dds_cli.user.**TokenFile**

Bases: object

A class to manage the saved token.

check_token_file_permissions()

Verify permissions for token file. Raises dds_cli.exceptions.DDSCLIException if permissions are not properly set.

Returns None otherwise.

delete_token()

Deletes the token file.

file_exists()

Returns True if the token file exists.

read_token()

Attempts to fetch a valid token from the token file.

Returns None if no valid token can be found.

save_token(*token*)

Saves the token to the token file.

token_expired()

Check how old the token is based on the modification time of the token file.

It compares the age with the dds variables TOKEN_MAX_AGE and TOKEN_WARNING_AGE to decide what to do.

Returns True if the token has expired, False otherwise.

** Notice, this does not actually know the expiration time of the token, it's possible that the expiration time has changed upstream in which case the variables need to be updated.

if check is True, token age will be reported to stdout

token_report()

Produce report of token status.

class dds_cli.user.**User**(*username: str*, *force_renew_token: bool = False*, *no_prompt: bool = False*)

Bases: object

Represents a DDS user.

when instantiating, an authentication token will be read from a file or renewed from the DDS API if the saved token is not found or has expired.

property token_dict

dds_cli.utils

DDS CLI utils module.

`dds_cli.utils.`**`calculate_magnitude`**(*projects*, *keys*, *iec_standard=False*)

Calculate magnitude of values.

Uses the project list, obtains the values assigned to a particular key iteratively and calculates the best magnitude to format this set of values consistently.

`dds_cli.utils.`**`format_api_response`**(*response*, *key*, *magnitude=None*, *iec_standard=False*)

Take a value e.g. bytes and reformat it to include a unit prefix.

`dds_cli.utils.`**`get_json_response`**(*response*)

Get json output from requests response.

Module contents

DDS CLI.

`class` `dds_cli.`**`DDSEndpoint`**

Bases: `object`

Defines all DDS urls.

`BASE_ENDPOINT = 'http://127.0.0.1:5000/api/v1'`

`BASE_ENDPOINT_LOCAL = 'http://127.0.0.1:5000/api/v1'`

`BASE_ENDPOINT_REMOTE = 'https://dds.dckube.scilifelab.se/api/v1'`

`CREATE_PROJ = 'http://127.0.0.1:5000/api/v1/proj/create'`

`ENCRYPTED_TOKEN = 'http://127.0.0.1:5000/api/v1/user/encrypted_token'`

`FILE_INFO = 'http://127.0.0.1:5000/api/v1/file/info'`

`FILE_INFO_ALL = 'http://127.0.0.1:5000/api/v1/file/all/info'`

`FILE_MATCH = 'http://127.0.0.1:5000/api/v1/file/match'`

`FILE_NEW = 'http://127.0.0.1:5000/api/v1/file/new'`

`FILE_UPDATE = 'http://127.0.0.1:5000/api/v1/file/update'`

`INVOICE = 'http://127.0.0.1:5000/api/v1/invoice'`

`LIST_FILES = 'http://127.0.0.1:5000/api/v1/files/list'`

`LIST_PROJ = 'http://127.0.0.1:5000/api/v1/proj/list'`

`LIST_PROJ_USERS = 'http://127.0.0.1:5000/api/v1/proj/users'`

`PROJ_PRIVATE = 'http://127.0.0.1:5000/api/v1/proj/private'`

`PROJ_PUBLIC = 'http://127.0.0.1:5000/api/v1/proj/public'`

`REMOVE_FILE = 'http://127.0.0.1:5000/api/v1/file/rm'`

`REMOVE_FOLDER = 'http://127.0.0.1:5000/api/v1/file/rmdir'`

`REMOVE_PROJ_CONT = 'http://127.0.0.1:5000/api/v1/proj/rm'`

`S3KEYS = 'http://127.0.0.1:5000/api/v1/s3/proj'`

`TIMEOUT = 5`

USAGE = 'http://127.0.0.1:5000/api/v1/usage'

USER_ADD = 'http://127.0.0.1:5000/api/v1/user/add'

class dds_cli.**FileSegment**

Bases: object

Defines information on signatures, file chunks, etc.

DDS_SIGNATURE = b'DelSys'

SEGMENT_SIZE_CIPHER = 65552

SEGMENT_SIZE_RAW = 65536

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX