

MyProject Documentation

Contents

Module dabapush	4
Using dabapush	4
First steps	4
Command Reference	5
Invocation Pattern:	5
Commands:	5
Programmatic documention	6
Extending dabapush and developers guide	6
dabapush's Approach to Projects and Configuration	6
Class References	6
Developer Installation	7
Sub-modules	7
Module dabapush.Configuration	7
Sub-modules	7
Module dabapush.Configuration.ConfigurationError	8
Classes	8
Class ConfigurationError	8
Ancestors (in MRO)	8
Module dabapush.Configuration.FileWriterConfiguration	8
Classes	8
Class FileWriterConfiguration	8
Ancestors (in MRO)	8
Descendants	8
Methods	8
Parameters	9
Returns	9
Parameters	9
Returns	9
Module dabapush.Configuration.PlugInConfiguration	9
Classes	9
Class PlugInConfiguration	9
Ancestors (in MRO)	9
Descendants	9
Class variables	9
Static methods	9
Module dabapush.Configuration.ProjectConfiguration	10
Classes	10
Class ProjectConfiguration	10
Parameters	10
Returns	10
Ancestors (in MRO)	10

Class variables	10
Methods	10
Parameters	11
Returns	11
Returns	11
Returns	11
Returns	11
Parameters	12
Returns	12
Parameters	12
Returns	12
Parameters	12
Returns	12
Module dabapush.Configuration.ReaderConfiguration	12
Classes	12
Class ReaderConfiguration	12
Ancestors (in MRO)	12
Descendants	13
Class variables	13
Module dabapush.Configuration.Registry	13
Classes	13
Class Registry	13
Ancestors (in MRO)	13
Class variables	13
Static methods	13
Parameters	13
Returns	13
Parameters	13
Returns	14
Methods	14
Parameters	14
Returns	14
Parameters	14
Returns	14
Parameters	14
Returns	15
Parameters	15
Returns	15
Module dabapush.Configuration.WriterConfiguration	15
Classes	15
Class WriterConfiguration	15
Ancestors (in MRO)	15
Descendants	15
Class variables	15
Module dabapush.Dabapush	15
Classes	15
Class Dabapush	15
Parameters	15
Returns	16
Methods	16
Returns	16
Returns	17
Parameters	17
Returns	17

Module dabapush.Reader	18
Sub-modules	18
Module dabapush.Reader.NDJSONReader	18
Classes	18
Class NDJSONReader	18
Ancestors (in MRO)	18
Methods	18
Class NDJSONReaderConfiguration	19
Ancestors (in MRO)	19
Class variables	19
Methods	19
Module dabapush.Reader.Reader	19
Classes	19
Class Reader	19
Parameters	19
Returns	19
Ancestors (in MRO)	19
Descendants	19
Methods	19
Module dabapush.Reader.TwacpicReader	20
Classes	20
Class TwacpicReader	20
Ancestors (in MRO)	20
Methods	20
Class TwacpicReaderConfiguration	20
Ancestors (in MRO)	20
Class variables	20
Methods	20
Module dabapush.Writer	21
Sub-modules	21
Module dabapush.Writer.CSVWriter	21
Classes	21
Class CSVWriter	21
Ancestors (in MRO)	21
Methods	21
Class CSVWriterConfiguration	21
Ancestors (in MRO)	21
Class variables	21
Instance variables	22
Methods	22
Module dabapush.Writer.DBWriter	22
Classes	22
Class DBWriter	22
Ancestors (in MRO)	22
Methods	22
Parameters	22
Returns	22
Module dabapush.Writer.NDJSONWriter	22
Classes	22
Class NDJSONWriter	22
Ancestors (in MRO)	22
Methods	23

Class NDJSONWriterConfiguration	23
Ancestors (in MRO)	23
Class variables	23
Methods	23
Module dabapush.Writer.Writer	23
Classes	23
Class Writer	23
Descendants	23
Instance variables	23
Methods	24
Parameters	24
Returns	24
Module dabapush.create_subcommand	24
Module dabapush.discover_subcommand	24
Module dabapush.main	24
Module dabapush.reader_subcommand	24
Module dabapush.run_subcommand	24
Module dabapush.update_subcommand	24
Module dabapush.utils	24
Functions	24
Function flatten	24
Function join	25
Returns	25
Function safe_access	25
Parameters	25
Returns	25
Module dabapush.writer_subcommand	25

Module dabapush

Database pusher for social media data (Twitter for the beginning) – pre-alpha version

Using dabapush

[dabapush](#) is a tool to read longer running data collections and write them to another file format or persist them into a database. It is designed to run periodically, e.g. controlled by `cron`, thus, for convenience of use project-based configurations which contain all required information on what to read where and what to do with it. A **project** may have one or more **jobs**, each job consists of a reader and a writer configuration, e.g. read JSON-files from the Twitter API that we stored in folder `/home/user/fancy-project/twitter/` and write the flattened and compiled data set in to `/some/where/else` as CSV files.

First steps

In order to run a first [dabapush](#)-job we'll need to create a project configuration. This is done by calling:

```
dabapush create
```

By default this walks you through the configuration process in a step-by-step manner. Alternatively, you could call:

```
dabapush create --non-interactive
```

This will create an empty configuration, you'll have to fill out the required information by e.g. calling:

```
dabapush reader add NDJSON default
dabapush writer add CSV default
```

Whereas reader add/writer add is the verb, NDJSON or CSV is the plugin to add and default is the pipeline name.

Of course you can edit the configuration after creation in your favorite editor, but **BEWARE NOT TO TEMPER WITH THE YAMI-TAGS!!!**.

Although,

To run the newly configured job, please call:

```
dabapush run default
```

Command Reference

Invocation Pattern:

```
dabapush <command> <subcommand?> <options>
```

Commands:

create - creates a dabapush project (invokes interactive prompt)

Options:

--non-interactive, create an empty configuration and exit

--interactive, *this is the default behavior*: prompts for user input on

- project name,
- project authors name,
- project author email address(es) for notifications
- manually configure targets or run discover?

run all - collect all known items and execute targets/destinations

run <target> - run a single writer and/or named target

Options:

--force-rerun, -r: forces all data to be read, ignores already logged data

update - update items for target(s)

update <target> or update all - update the target's files-to-be-read list

reader - interact with readers

reader configure <name> - configure the reader for one or more subproject(s); Reader configuration is inherited from global to local level; throws if configuration is incomplete and defaults are missing

reader list: returns a table of all configured readers, with

reader list_all: returns a table of all registered reader plugins
reader add <type> <name>: add a reader to the project configuration
Options:
--input-directory <path>: directory to be read
--pattern <pattern>: pattern for matching file names against.
remove <name>: remove a reader from the project configuration.
register <path>: not there yet

discover - discover (possible) targets in project directory and configure them automatically
- yeah, you dream of that, don't you?

writer - interact with writers

writer add <type> <name>:

writer remove <name>: removes the writer for the given name

writer list - returns table of all writers, with

writer list_all: returns a table of all registered writer plugins

writer configure <name> or writer configure all

Options:

--output-dir, -o <path>: default for all targets: <project-dir>/output/<target-name>

--output-pattern, -p <pattern>: pattern used for file name creation e.g. 'YYYY-MM-dd', file extension is added by the writer and cannot be overwritten

--roll-over, -r:

--roll-over, -r :

--roll-over -r <None>: should be the output chunked? Give either a file-size or a number of lines for roll-over or None to disable chunking

writer register <path> <class-name>: something for the roadmap.

deregister <name> <registry-name>: something for the roadmap.

Programmatic documentation

Extending dabapush and developers guide

dabapush's Approach to Projects and Configuration

Class References

Dabapush

Reader

Writer

Configuration | |

```
-----|---|
      ||
      ||
      ||
      ||
      ||
```

Developer Installation

1. Install poetry¹.
2. Clone repository.
3. In the cloned repository's root directory run poetry install.
4. Run poetry shell to start development virtualenv.
5. Run [dabapush](#) create to create your first project.
6. Run pytest to run all tests. ## 7. Run `pdocr --http localhost:8080 dabapush` to locally serve this documentation.

Contact: smo (ät) leibniz-hbi.de

Sub-modules

- [dabapush.Configuration](#)
- [dabapush.Dabapush](#)
- [dabapush.Reader](#)
- [dabapush.Writer](#)
- [dabapush.create_subcommand](#)
- [dabapush.discover_subcommand](#)
- [dabapush.main](#)
- [dabapush.reader_subcommand](#)
- [dabapush.run_subcommand](#)
- [dabapush.update_subcommand](#)
- [dabapush.utils](#)
- [dabapush.writer_subcommand](#)

Module `dabapush.Configuration`

Sub-modules

- [dabapush.Configuration.ConfigurationError](#)

¹<https://python-poetry.org/docs/#installation>

- [dabapush.Configuration.FileWriterConfiguration](#)
- [dabapush.Configuration.PlugInConfiguration](#)
- [dabapush.Configuration.ProjectConfiguration](#)
- [dabapush.Configuration.ReaderConfiguration](#)
- [dabapush.Configuration.Registry](#)
- [dabapush.Configuration.WriterConfiguration](#)

Module `dabapush.Configuration.ConfigurationError`

Classes

Class `ConfigurationError`

```
class ConfigurationError(
    *args: object
)
```

Ancestors (in MRO)

- [builtins.Exception](#)
- [builtins.BaseException](#)

Module `dabapush.Configuration.FileWriterConfiguration`

Classes

Class `FileWriterConfiguration`

```
class FileWriterConfiguration(
    name,
    id=None,
    chunk_size: int = 2000,
    path: str = '.',
    name_template: str = '${date}_${time}_${name}.${type}'
)
```

Abstract class describing configuration items for a file based writer

Ancestors (in MRO)

- [dabapush.Configuration.WriterConfiguration.WriterConfiguration](#)
- [dabapush.Configuration.PlugInConfiguration.PlugInConfiguration](#)
- [yaml.YAMLObject](#)

Descendants

- [dabapush.Writer.CSVWriter.CSVWriterConfiguration](#)
- [dabapush.Writer.NDJSONWriter.NDJSONWriterConfiguration](#)

Methods

Method `make_file_name`

```
def make_file_name(
    self,
    additional_keys: dict = {}
) -> str
```


Parameters

additional_keys : dict: (Default value = {})
additional_keys : dict : : (Default value = {})

Returns

Method set_name_template

```
def set_name_template(  
    self,  
    template: str  
)
```

Parameters

template : str: template : str: template : str : :

Returns

Module dabapush.Configuration.PlugInConfiguration

Classes

Class PlugInConfiguration

```
class PlugInConfiguration(  
    name: str,  
    id: str  
)
```

Ancestors (in MRO)

- [yaml.YAMLObject](#)

Descendants

- [dabapush.Configuration.ReaderConfiguration.ReaderConfiguration](#)
- [dabapush.Configuration.WriterConfiguration.WriterConfiguration](#)

Class variables

Variable yaml_tag

Static methods

Method get_instance

```
def get_instance() -> Reader
```

Module `dabapush.Configuration.ProjectConfiguration`

Classes

Class `ProjectConfiguration`

```
class ProjectConfiguration(  
    readers: Dict[str, dabapush.Configuration.ReaderConfiguration.ReaderConfiguration] = {},  
    writers: Dict[str, dabapush.Configuration.WriterConfiguration.WriterConfiguration] = {},  
    author: str = '',  
    name: str = ''  
)
```

`ProjectConfiguration` hold necessary configuration informations

A `ProjectConfiguration` is for reading and writing data as well as the project's meta data e.g. author name(s) and email addresses.

Parameters

Returns

Initialize a `ProjectConfiguration` with optional reader and/or writer dicts

Ancestors (in MRO)

- [yaml.YAMLObject](#)

Class variables

Variable `yaml_tag`

Methods

Method `add_reader`

```
def add_reader(  
    self,  
    type: str,  
    name: str  
) -> None
```

add a reader configuration to the project

Parameters

`type` : str: registry of the configuration to add `name` : str: name of the configuration to add

Returns: Nothing. `type` : str: `name` : str: `type` : str: `name` : str: `type` : str :

`name` : str :

Returns

Raises

ConfigurationError if no local or global configurations are found

Method add_writer

```
def add_writer(
    self,
    type: str,
    name: str
) -> None
```

Parameters

type : str: name : str: Returns: None: nothing to see, carry on. type : str: name : str: type : str: name : str: type : str :

name : str :

Returns**Method** list_readers

```
def list_readers(
    self
) -> List[dict]
```

list all configured readers

Returns: List[Dict]: list of dicts with name- and id-fields

Parameters

Returns**Method** list_writers

```
def list_writers(
    self
)
```

list all configured writers

Returns: List[Dict]: list of dicts with name- and id-fields

Parameters

Returns**Method** remove_reader

```
def remove_reader(
    self,
    name: str
) -> None
```

remove a reader from the configuration

Parameters

name : str: name of the reader to be removed Returns: Nada. name : str: name : str: name : str :

Returns**Method** remove_writer

```
def remove_writer(
    self,
    name: str
)
```

Parameters

name : str: name : str: name : str: name : str : :

Returns

Method set_author

```
def set_author(
    self,
    author
)
```

Parameters

author :

Returns

Method set_name

```
def set_name(
    self,
    name
)
```

Parameters

name :

Returns

Module `dabapush.Configuration.ReaderConfiguration`

Classes

Class `ReaderConfiguration`

```
class ReaderConfiguration(
    name,
    id,
    read_path: str,
    pattern: str
)
```

Ancestors (in MRO)

- [dabapush.Configuration.PluginConfiguration.PluginConfiguration](#)
- [yaml.YAMLObject](#)

Descendants

- [dabapush.Reader.NDJSONReader.NDJSONReaderConfiguration](#)
- [dabapush.Reader.TwacpicReader.TwacpicReaderConfiguration](#)

Class variables

Variable `yaml_tag`

Module `dabapush.Configuration.Registry`

Classes

Class `Registry`

```
class Registry(  
    readers: Dict[str, str] = {},  
    writers: Dict[str, str] = {}  
)
```

Ancestors (in MRO)

- [yaml.YAMLObject](#)

Class variables

Variable `yaml_tag`

Static methods

Method `get_reader`

```
def get_reader(  
    type: str  
) -> dabapush.Configuration.ReaderConfiguration.ReaderConfiguration
```

Parameters

`type : str`: registry key Returns: `ReaderConfiguration` or `None`: the requested `ReaderConfiguration` or `None` if no matching configuration is found. `type : str`: `type : str`: `type : str` :

Returns

Method `get_writer`

```
def get_writer(  
    type: str  
) -> dabapush.Configuration.WriterConfiguration.WriterConfiguration
```

Parameters

`type : str`: `type : str`: `type : str`: `type : str` :

Returns

Method `list_all_readers`

```
def list_all_readers() -> List[str]
```

Method `list_all_writers`

```
def list_all_writers() -> List[str]
```

Methods

Method `list_writers`

```
def list_writers(  
    self  
) -> List[str]
```

Method `register_reader`

```
def register_reader(  
    self,  
    name: str,  
    plugin_configuration  
) -> None
```

Parameters

`name : str`: constructor : param `name: str`: `plugin_configuration : param name: str`: `name : str : :`

Returns

Method `register_writer`

```
def register_writer(  
    self,  
    name: str,  
    constructor  
) -> None
```

Parameters

`name : str`: constructor : param `name: str`: `name : str`: `name : str : :`

Returns

Method `remove_reader`

```
def remove_reader(  
    self,  
    name: str  
) -> bool
```

Parameters

`name : str`: `name : str`: `name : str`: `name : str : :`

Returns

Method `remove_writer`

```
def remove_writer(  
    self,  
    name: str  
) -> bool
```

Parameters

name : str: name : str: name : str: name : str : :

Returns

Module `dabapush.Configuration.WriterConfiguration`

Classes

Class `WriterConfiguration`

```
class WriterConfiguration(  
    name,  
    id=None,  
    chunk_size: int = 2000  
)
```

Ancestors (in MRO)

- [dabapush.Configuration.PluginConfiguration.PluginConfiguration](#)
- [yaml.YAMLObject](#)

Descendants

- [dabapush.Configuration.FileWriterConfiguration.FileWriterConfiguration](#)

Class variables

Variable `yaml_tag`

Module `dabapush.Dabapush`

Classes

Class `Dabapush`

```
class Dabapush(  
    install_dir: pathlib.Path = PosixPath('/home/philippkessling/repos/dabapush'),  
    working_dir: pathlib.Path = PosixPath('/home/philippkessling/repos/dabapush')  
)
```

This is the main class for this application.

It is a Singleton pattern class and follows the interface pattern as well.

Parameters

Returns

Methods

Method `gc_load`

```
def gc_load(  
    self  
)
```

load the global registry and configuration

Method `jb_run`

```
def jb_run(  
    self,  
    targets: list  
)
```

runs the job(s) configured in the current directory

Parameters

`targets : list[str]:` targets : list[str]: `targets : list[str] : :`

Returns

Method `jb_update`

```
def jb_update(  
    self  
)
```

update the current job's targets

Method `pr_init`

```
def pr_init(  
    self  
)
```

Initialize a new project in the current directory

Method `pr_read`

```
def pr_read(  
    self  
) -> bool
```

Read the project configuration file in the current directory

Parameters

Returns

type bool Indicates whether loading load successful

Method `pr_write`

```
def pr_write(  
    self  
)
```

Write the current configuration to the project configuration file in the current directory

Method `rd_add`

```
def rd_add(  
    self,  
    reader: str,  
    name: str  
)
```

add a reader to the current project

Parameters

`reader : str: name : str: reader : str: name : str: reader : str : :`
`name : str :`

Returns**Method** `rd_list`

```
def rd_list(  
    self  
)
```

Lists all available readers

Method `rd_rm`

```
def rd_rm(  
    self  
)
```

remove a reader from the current configuration

Method `rd_update`

```
def rd_update(  
    self  
)
```

update a reader's configuration

Method `update_reader_targets`

```
def update_reader_targets(  
    self,  
    name: str  
) -> None
```

Parameters

`name : str: name : str: name : str : :`

Returns**Method** `wr_add`

```
def wr_add(  
    self  
)
```

add a reader to the current project

Method `wr_list`

```
def wr_list(  
    self  
)
```

Lists all available readers

Method `wr_rm`

```
def wr_rm(  
    self  
)
```

remove a reader from the current configuration

Method `wr_update`

```
def wr_update(  
    self  
)
```

update a reader's configuration

Module `dabapush.Reader`**Sub-modules**

- [dabapush.Reader.NDJSONReader](#)
- [dabapush.Reader.Reader](#)
- [dabapush.Reader.TwacpicReader](#)

Module `dabapush.Reader.NDJSONReader`**Classes****Class** `NDJSONReader`

```
class NDJSONReader(  
    config: NDJSONReaderConfiguration  
)
```

Ancestors (in MRO)

- [dabapush.Reader.Reader.Reader](#)
- [abc.ABC](#)

Methods**Method** `read`

```
def read(  
    self  
) -> Generator[<built-in function any>, <built-in function any>, <built-in function any>]
```

reads multiple ndjson files and emits them line by line

Class NDJSONReaderConfiguration

```
class NDJSONReaderConfiguration(
    name,
    id=None,
    read_path: str = '.',
    pattern: str = '*.ndjson',
    flatten_dicts=True
)
```

Ancestors (in MRO)

- [dabapush.Configuration.ReaderConfiguration.ReaderConfiguration](#)
- [dabapush.Configuration.PluginConfiguration.PluginConfiguration](#)
- [yaml.YAMLObject](#)

Class variables

Variable `yaml_tag`

Methods

Method `get_instance`

```
def get_instance(
    self
) -> dabapush.Reader.NDJSONReader.NDJSONReader
```

Module `dabapush.Reader.Reader`

Classes

Class `Reader`

```
class Reader(
    config: dabapush.Configuration.ReaderConfiguration.ReaderConfiguration
)
```

Abstract base class for all reader plugins. BEWARE: readers and writers are never to be instanced directly by the user but rather will be obtain by calling `get_instance()` on their specific Configuration-counterparts.

Parameters

Returns

Ancestors (in MRO)

- [abc.ABC](#)

Descendants

- [dabapush.Reader.NDJSONReader.NDJSONReader](#)
- [dabapush.Reader.TwacpicReader.TwacpicReader](#)

Methods

Method read

```
def read(  
    self  
)
```

Module `dabapush.Reader.TwacpicReader`

Classes

Class `TwacpicReader`

```
class TwacpicReader(  
    config: ReaderConfiguration  
)
```

Reader to read ready to read Twitter json data

Ancestors (in MRO)

- [dabapush.Reader.Reader.Reader](#)
- [abc.ABC](#)

Methods

Method read

```
def read(  
    self  
) -> Generator[<built-in function any>, <built-in function any>, <built-in function any>]
```

reads the configured path a returns a generator of single posts

Class `TwacpicReaderConfiguration`

```
class TwacpicReaderConfiguration(  
    name,  
    id=None,  
    read_path: str = None,  
    pattern: str = '*.json'  
)
```

Reader configuration for reading Twacpic's Twitter JSON files.

Ancestors (in MRO)

- [dabapush.Configuration.ReaderConfiguration.ReaderConfiguration](#)
- [dabapush.Configuration.PluginConfiguration.PluginConfiguration](#)
- [yaml.YAMLObject](#)

Class variables

Variable `yaml_tag`

Methods

Method `get_instance`

```
def get_instance(  
    self  
) -> dabapush.Reader.TwacpicReader.TwacpicReader
```

Module `dabapush.Writer`

Sub-modules

- [dabapush.Writer.CSVWriter](#)
- [dabapush.Writer.DBWriter](#)
- [dabapush.Writer.NDJSONWriter](#)
- [dabapush.Writer.Writer](#)

Module `dabapush.Writer.CSVWriter`

Classes

Class `CSVWriter`

```
class CSVWriter(  
    config: CSVWriterConfiguration  
)
```

Ancestors (in MRO)

- [dabapush.Writer.Writer.Writer](#)

Methods

Method `persist`

```
def persist(  
    self  
)
```

Class `CSVWriterConfiguration`

```
class CSVWriterConfiguration(  
    name,  
    id=None,  
    chunk_size: int = 2000,  
    path: str = '.',  
    name_template: str = '${date}_${time}_${name}_${chunk_number}.${type}'  
)
```

Ancestors (in MRO)

- [dabapush.Configuration.FileWriterConfiguration.FileWriterConfiguration](#)
- [dabapush.Configuration.WriterConfiguration.WriterConfiguration](#)
- [dabapush.Configuration.PluginConfiguration.PluginConfiguration](#)
- [yaml.YAMLObject](#)

Class variables

Variable `yaml_tag`

Instance variables

Variable `file_path` Type: `pathlib.Path`

Methods

Method `get_instance`

```
def get_instance(  
    self  
)
```

Module `dabapush.Writer.DBWriter`

Classes

Class `DBWriter`

```
class DBWriter
```

Ancestors (in MRO)

- [dabapush.Writer.Writer.Writer](#)

Methods

Method `persist`

```
def persist(  
    self,  
    chunkSize: int  
)
```

Parameters

`chunkSize : int`: `chunkSize : int`: `chunkSize : int` : :

Returns

Module `dabapush.Writer.NDJSONWriter`

Classes

Class `NDJSONWriter`

```
class NDJSONWriter(  
    config: NDJSONWriterConfiguration  
)
```

Ancestors (in MRO)

- [dabapush.Writer.Writer.Writer](#)

Methods

Method `persist`

```
def persist(  
    self  
)
```

Class `NDJSONWriterConfiguration`

```
class NDJSONWriterConfiguration(  
    name,  
    id=None,  
    chunk_size: int = 2000,  
    path: str = '.',  
    name_template: str = '${date}_${time}_${name}.${type}'  
)
```

Ancestors (in MRO)

- [dabapush.Configuration.FileWriterConfiguration.FileWriterConfiguration](#)
- [dabapush.Configuration.WriterConfiguration.WriterConfiguration](#)
- [dabapush.Configuration.PluginConfiguration.PluginConfiguration](#)
- [yaml.YAMLObject](#)

Class variables

Variable `yaml_tag`

Methods

Method `get_instance`

```
def get_instance(  
    self  
)
```

Module `dabapush.Writer.Writer`

Classes

Class `Writer`

```
class Writer(  
    config: dabapush.Configuration.WriterConfiguration.WriterConfiguration  
)
```

Descendants

- [dabapush.Writer.CSVWriter.CSVWriter](#)
- [dabapush.Writer.DBWriter.DBWriter](#)
- [dabapush.Writer.NDJSONWriter.NDJSONWriter](#)

Instance variables

Variable id

Variable name

Methods

Method persist

```
def persist(  
    self  
) -> None
```

Method write

```
def write(  
    self,  
    queue: Generator[<built-in function any>, <built-in function any>, <built-in function any>]  
)
```

Parameters

df : dict queue : Generator[any: any : param any]: queue : Generator[any: queue
: Generator[any : :
any] :

Returns

Module dabapush.create_subcommand

Module dabapush.discover_subcommand

Module dabapush.main

Module dabapush.reader_subcommand

Module dabapush.run_subcommand

Some module documentation

Module dabapush.update_subcommand

Module dabapush.utils

Functions

Function flatten

```
def flatten(  
    thing: dict,  
    namespace: str = None,  
    sep: str = '.'  
) -> dict
```


Flattens a nested array, flattened keys are joined together with the specified separator.

Parameters

thing : dict: Nested dict to flatten namespace : str: (Default value = None) sep : str: (Default value = '.') thing : dict : :

namespace : str : (Default value = None)

sep : str : (Default value = '.')

Returns

type dict: the flattened dict

Function join

```
def join(
    id: str,
    includes: list,
    id_key: str
) -> <built-in function any>
```

looks up an entity in a array of dicts by given key.

Parameters

id : str: includes : list[any]: id_key : str: id : str: includes : list[any]: id_key : str: id : str : :

includes : list[any] :

id_key : str :

Returns

Function safe_access

```
def safe_access(
    thing: dict,
    path: list
)
```

Parameters

thing : dict: path : list[str]: thing : dict: path : list[str]: thing : dict : :

path : list[str] :

Returns

Module dabapush.writer_subcommand

Generated by *pdoc* 0.10.0 (<https://pdoc3.github.io>).