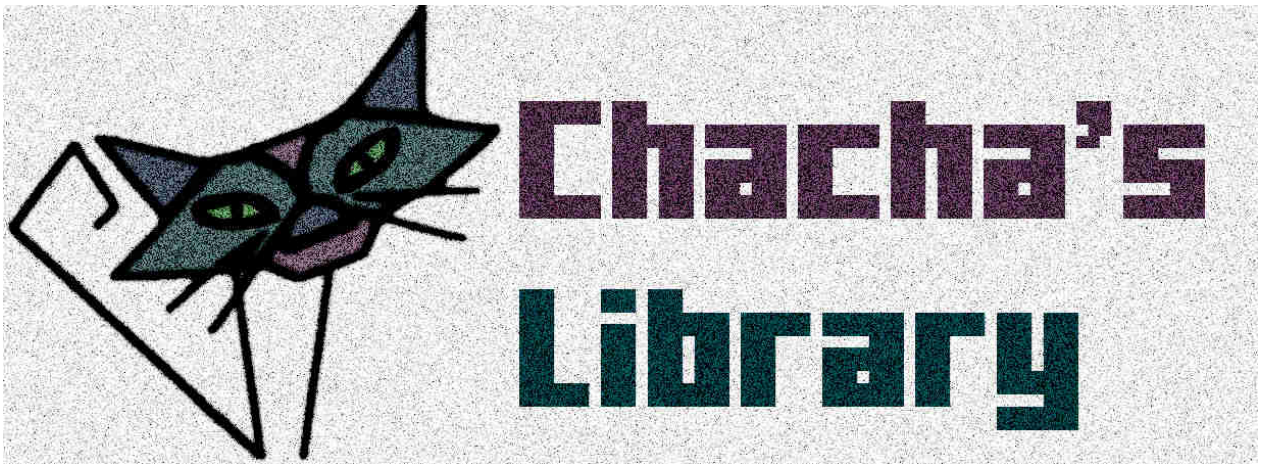
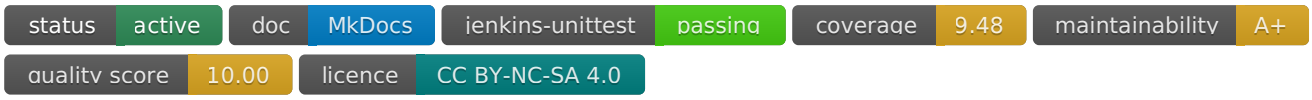


dabdatasync

User Manual

chacha

Table of contents



1. Python project template

A nice template to start blank python projets.

This template automate a lot of handy things and allow CI/CD automatic releases generation.

It is also collectings data to feed Jenkins build.

Checkout Latest Documentation.

1.1 Features

1.1.1 Generic pipeline skeleton:

```
- Prepare
- GetCode
- BuildPackage
- Install
- CheckCode
- PlotMetrics
- RunUnitTests
- GenDOC
- PostRelease
```

1.1.2 CI/CD Environment

```
- Jenkins
- Gitea (with patch for dynamic Readme variables: https://chacha.ddns.net/gitea/chacha/GiteaMarkupVariable)
- Docker
- MkDocsWeb
```

1.1.3 CI/CD Helper libs

```
- VirtualEnv
- Changelog generation based on commits
- copier
- pylint + pylint_json2html
- mypy
- unittest + xmlrunner + junitparser + junit2htmlreport
- mkdocs
```

1.1.4 Python project

```
- Full .toml implementation
- .whl automatic generation
- dynamic versionning using git repository
- embedded unit-test
```

2. Usage

2.1 Pulvinar dolor

Donec dapibus est fermentum justo volutpat condimentum. Integer quis nunc neque. Donec dictum vehicula justo, in facilisis ex tincidunt in. Vivamus sollicitudin sem dui, id mollis orci facilisis ut. Proin sed pulvinar dolor. Donec volutpat commodo urna imperdiet pulvinar. Fusce eget aliquam risus. Vivamus viverra luctus ex, in finibus mi. Nullam elementum dapibus mollis. Ut suscipit volutpat ex, quis feugiat lacus consectetur eu.

2.2 Condimentum faucibus

Quisque auctor egestas sem, luctus suscipit ex maximus vitae. Duis facilisis augue et condimentum faucibus. Donec cursus, enim a sagittis egestas, lectus lorem eleifend libero, at tincidunt leo magna at libero. Nunc eros velit, suscipit luctus tempor vel, finibus et est. Curabitur efficitur pretium pulvinar. Donec urna lectus, vulputate quis turpis sed, placerat congue urna. Phasellus aliquet fermentum quam, non auctor elit porta nec. Morbi eu ligula at nisl ultricies condimentum vitae id ante.

2.3 Aliquam lacinia

In volutpat lorem ex, et fringilla nibh faucibus quis. Mauris et arcu elementum, auctor dui vitae, egestas arcu. Duis sit amet aliquam quam. Phasellus a odio turpis. Etiam tristique mi eu enim varius, eget facilisis est vestibulum. Aliquam lacinia nec purus sed luctus. Cras at laoreet erat.

3. Reference

3.1 Dabdatasync

3.1.1 metadata

Metadata module module

Attributes

__NAME__ MODULE-ATTRIBUTE

```
__Name__ = metadata['Name']
```

__SUMMUARY__ MODULE-ATTRIBUTE

```
__Summuary__ = metadata['Summary']
```

__VERSION__ MODULE-ATTRIBUTE

```
__version__ = version('dabdatasync')
```

DIST MODULE-ATTRIBUTE

```
dist = distribution('dabdatasync')
```

3.1.2 Compressors

Compressor interface and implementation

Classes

A_DATASYNC_COMPRESSOR

Bases: `ABC`

abstract compressor class

Attributes

compressor_name `class-attribute` `instance-attribute`

```
compressor_name: str = 'ABSTRACT'
```

suffix `instance-attribute`

```
suffix: str
```

Functions

compress `classmethod`

```
compress(path_in: Path, file_out: IO)
```

compress method

uncompress `classmethod`

```
uncompress(path_in: Path, path_out: Path)
```

uncompress method

DATASYNC_COMPRESSOR__TAR_GZ

Bases: `A_DataSync_Compressor`

Concrete compressor class - .tar.gz compressor

Attributes

compressor_name `class-attribute` `instance-attribute`

```
compressor_name: str = 'tar_gz'
```

suffix `class-attribute` `instance-attribute`

```
suffix: str = '.tar.gz'
```

DATASYNC_COMPRESSOR__TAR_LZ4

Bases: `A_DataSync_Compressor`

Concrete compressor class - .tar.lz4 compressor

Attributes

compressor_name `class-attribute` `instance-attribute`

```
compressor_name: str = 'tar_lz4'
```

suffix `class-attribute` `instance-attribute`

```
suffix: str = '.tar.lz4'
```

DATASYNC_COMPRESSORS

compressers container/factory class

Functions

get `classmethod`

```
get(compressor_name: str) -> type[A_DataSync_Compressor]
```

get a specific compressor

get_list `classmethod`

```
get_list() -> list[str]
```

return the available compressor name list

register `classmethod`

```
register(_cls: type[A_DataSync_Compressor]) -> type[A_DataSync_Compressor]
```

register a new compressor

3.1.3 Datasync

Nextcloud abstract interface

Classes

A_DATASYNC

```
A_DataSync(manifest: dict[str, Any], cls_compressor: type[A_DataSync_Compressor])
```

Bases: `ABC`

Abstract DataSync class

Attributes

connected `instance-attribute`

```
connected: bool = False
```

priority `class-attribute` `instance-attribute`

```
priority: int = 0
```

service_name `class-attribute` `instance-attribute`

```
service_name: str = 'ABSTRACT'
```

Functions

configure

```
configure() -> None
```

configure the class instance

connect

```
connect() -> None
```

connect to the service

get_datasync_records

```
get_datasync_records() -> list[A_DataSync_Record]
```

get list of records

pull_data

```
pull_data() -> None
```

pull data from the service

push_data

```
push_data() -> None
```

push data to the service

set_compressor

```
set_compressor(compressor_name: str) -> None
```

set compressor to be used

test_applicable `classmethod`

```
test_applicable(manifest: dict[str, Any]) -> bool
```

check if a concrete class is applicable - generic

try_get_instance `classmethod`

```
try_get_instance(manifest: dict[str, Any], cls_compressor: type[A_DataSync_Compressor]) -> Self | None
```

try to get an instance of a concrete class

wipe_local_data

```
wipe_local_data() -> None
```

wipe local data

wipe_remote_data

```
wipe_remote_data() -> None
```

wipe data on the service

DATASYNC_FACTORY

DataSync Factory

Attributes

ar_cls_DataSync `class-attribute` `instance-attribute`

```
ar_cls_DataSync: set[type[A_DataSync]] = set()
```

cls_compressor `class-attribute` `instance-attribute`

```
cls_compressor: type[A_DataSync_Compressor] = DataSync_Compressor__tar_gz
```

manifest_path `class-attribute` `instance-attribute`

```
manifest_path: str = '/opt/pyDABFactoryAppliance/Manifest.json'
```

Functions

get_DataSync `classmethod`

```
get_DataSync() -> list[A_DataSync]
```

get and configure a DataSync Concrete class instance

get_list `classmethod`

```
get_list() -> list[str]
```

return the available DataSync concrete class name list

get_manifest_data `classmethod`

```
get_manifest_data() -> dict[str, Any]
```

tool method to get manifest

register `classmethod`

```
register(_cls: type[A_DataSync]) -> type[A_DataSync]
```

decorator to register a concrete class to the factory

3.1.4 Datasync nextcloud

Nextcloud datasync implementation

Classes

C_DATASYNC_NEXTCLOUD

```
C_DataSync_NextCloud(manifest: dict[Any, Any], cls_compressor: type[A_DataSync_Compressor])
```

Bases: `A_DataSync`

Concrete DataSync class - Nextcloud

Attributes

client `instance-attribute`

```
client: Client
```

connected `instance-attribute`

```
connected: bool = False
```

nextcloud_address `instance-attribute`

```
nextcloud_address: str
```

nextcloud_password `instance-attribute`

```
nextcloud_password: str
```

nextcloud_path `instance-attribute`

```
nextcloud_path: str
```

nextcloud_user `instance-attribute`

```
nextcloud_user: str
```

priority `class-attribute` `instance-attribute`

```
priority: int = 100
```

service_name `class-attribute` `instance-attribute`

```
service_name: str = 'Nextcloud'
```

Functions

test_applicable `classmethod`

```
test_applicable(manifest: dict[str, Any]) -> bool
```

check if a concrete class is applicable - Nextcloud override

Functions

3.1.5 Exceptions

Exception declaration

Classes

DATASYNCEXCEPTION

Bases: `Exception`

generic datasync exception class

DATASYNCEXCEPTION_COMPRESSORNOTFOUND

Bases: `DataSyncException`

specific datasync exception class - Compressor Not Found

DATASYNCEXCEPTION_INVALIDMANIFEST

Bases: `DataSyncException`

specific datasync exception class - Dab appliance manifest not found

DATASYNCEXCEPTION_NOCONCRETERECORDCLASSFOUND

Bases: `DataSyncException`

specific datasync exception class - No concrete Record Class Found

DATASYNCEXCEPTION_NOVALIDSERVICEFOUND

Bases: `DataSyncException`

specific datasync exception class - Remote Valid Service Found

DATASYNCEXCEPTION_REMOTEDATANOTFOUND

Bases: `DataSyncException`

specific datasync exception class - Remote Data Not Found

DATASYNCEXCEPTION_SERVICENOTFOUND

Bases: `DataSyncException`

specific datasync exception class - Service Not Found

DATASYNCEXCEPTION_TOOMANYCOMPRESSORFOUND

Bases: `DataSyncException`

specific datasync exception class - Too Many Compressor Found

DATASYNCEXCEPTION_TOOMANYSERVICEFOUND

Bases: `DataSyncException`

specific datasync exception class - Too Many Service Found

3.1.6 Records

datasync records description and implementation

Classes

A_DATASYNC_RECORD

Bases: `BaseModel`, `ABC`

Abstract DataSync Record class

Attributes

name `instance-attribute`

```
name: str
```

rec_type `instance-attribute`

```
rec_type: str
```

value `instance-attribute`

```
value: str
```

Functions

compress `abstractmethod`

```
compress(cls_compressor: type[A_DataSync_Compressor], file_out: IO) -> None
```

compress the DataSync Record - virtual

uncompress `abstractmethod`

```
uncompress(cls_compressor: type[A_DataSync_Compressor], path_in: Path) -> None
```

uncompress the DataSync record - virtual

wipe `abstractmethod`

```
wipe()
```

wipe the record local data - virtual

C_DATASYNC_RECORD_FS

Bases: `A_DataSync_Record`

Concrete DataSync Record class - FileSystem

Attributes

path `class-attribute` `instance-attribute`

```
path: Optional[Path] = None
```

rec_type `class-attribute` `instance-attribute`

```
rec_type: str = 'fs'
```

Functions

compress

```
compress(cls_compressor: type[A_DataSync_Compressor], file_out: IO) -> None
```

compress the DataSync Record - concrete FS implementation

model_post_init

```
model_post_init(__context) -> None
```

uncompress

```
uncompress(cls_compressor: type[A_DataSync_Compressor], path_in: Path) -> None
```

uncompress the DataSync record - concrete FS implementation

wipe

```
wipe()
```

DATASYNC_RECORD_FACTORY

DataSync Record Factory

Attributes

ar_cls_DataSync_Record `class-attribute` `instance-attribute`

```
ar_cls_DataSync_Record: set[type[A_DataSync_Record]] = set()
```

Functions

get_C_DataSync_Record `classmethod`

```
get_C_DataSync_Record(name: str, rec_type: str, value: str) -> A_DataSync_Record | None
```

get a concrete DataSync Record class instance

register `classmethod`

```
register(_cls: type[A_DataSync_Record]) -> type[A_DataSync_Record]
```

decorator to register a concrete DataSync Record class

3.1.7 Utils

tools classes / functions

Functions

URLJOIN

```
urljoin(*args: str) -> str
```

Joins given arguments into an url. Trailing but not leading slashes are stripped for each argument.