Standard File Documentation

Release 0.0.0

Stephen Bunn

Aug 10, 2018

Contents

1 User Documentation **3**
1.1 Getting Started 3
1.2 Contributing 6

2 Project Reference **9**
2.1 Standard File Package 9

Python Module Index **15**

The popular Standard Notes provides a decent spec for basic text encryption called Standard File. Given that this spec doesn't have very many decent bindings for it in Python, I decided to take a stab at it.

CHAPTER 1

User Documentation

1.1 Getting Started

1.1.1 Installation and Setup

Installing the package should be easy through pipenv.

```
$ pipenv install standardfile
$ # or if you're old school...
$ pip install standardfile
```

Or you can build and install the package from the git repo.

```
$ git clone https://github.com/stephen-bunn/standardfile.git
$ cd ./standardfile
$ python setup.py install
```

1.1.2 Usage

Below is some documentation on how to do some basic things with this module.
Please keep in mind that this module is still very alpha and some things are definitely not supported and other things are probably broken.

Currently the way items are updated by the user is done poorly, I am working on making it easier to interact with items.

Logging In

There are two basic ways to login.

```
>>> from standardfile import User
>>> user = User('user@example.com')
>>> print(user.authenticated)
False
>>> user.authenticate('password')
>>> print(user.authenticated)
True
```

or...

```
>>> user = User.login('user@example.com', 'password')
>>> print(user.authenticated)
True
```

If you use multi-factor authentication then simply provide the `mfa` keyword argument with the current code.

```
>>> user = User.login('user@example.com', 'password')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/stephen-bunn/Git/standardfile/standardfile/user.py", line 228, in login
    user.authenticate(password, mfa=mfa)
  File "/home/stephen-bunn/Git/standardfile/standardfile/user.py", line 312, in␣
↪authenticate
    raise exceptions.MFARequired("mfa code is required but not provided")
standardfile.exceptions.MFARequired: mfa code is required but not provided
>>> user = User.login('user@example.com', 'password', mfa='123456')
>>> print(user.authenticated)
True
```

If you are not trying to connect to the default standard notes sync server, then you can supply the `host` keyword with the desired host.

```
>>> user = User.login('user@example.com', 'password', host='https://sync.example.com')
>>> print(user.authenticated)
True
```

Syncing Items

Standard File works by essentially cloning the server's synced directory. This is done through the simple `sync` command which clones the server items into the `user.sync_dir` folder.

```
>>> print(user.sync_dir)
/home/user/Git/standardfile/d79ca65d-3135-404c-a14d-e45b4226c101
>>> sync_results = user.sync()
>>> print(sync_results)
{'retrieved_items': [{'uuid': 'd3866137-f7f6-4b98-8b34-43cae416472e', 'content':
↪'002:32d60ba576f418baf1173527c2e9c0c82cd3642885cbd48c2dc86e30ed5dfaeb:d3866137-f7f6-
↪4b98-8b34-43cae416472e:9cab0dd683cac38b8fac8060a5d7f835:mrKYT+9jFsOuBO5Baa4jWA==',
↪'content_type': 'test', 'enc_item_key':
↪'002:681be79d198eab9fb57695b74a522af8169ecf75f14eed8b588bb4a4a45c4e3e:d3866137-f7f6-
↪4b98-8b34-43cae416472e:354d55bd7ac79d7a955372405a3c3a27:omJYXXy98pLj1JEGuSKB0/cc/
↪Wu9bnNa5SjLSKsz6DwOxBnRFesNCqIImSxL5omN98LU4a5iXhqYwRPYp833Bc4UY5/Fexn0eSATMqZ/tRM=
↪', 'auth_hash': None, 'created_at': '2018-06-07T23:08:48.023Z', 'updated_at': '2018-
↪06-07T23:44:05.369Z', 'deleted': False}], 'saved_items': [], 'unsaved': [], 'sync_
↪token': 'MjoxNTMwMTI4NjA2LjcwNTYyNDg=\n', 'cursor_token': None}
```

The retrieved items are accesible through the `user.items` dictionary.

```
>>> print(user.items)
{'d3866137-f7f6-4b98-8b34-43cae416472e': Item(uuid='d3866137-f7f6-4b98-8b34-
↪43cae416472e')}
```

The synced items will be accessible locally in the `user.sync_dir` directory as a file named `d3866137-f7f6-4b98-8b34-43cae416472e`.

Following the basic worflow of authentication based synced storage systems, *currently* we don't allow a `user` to access their items through `user.items` if they are not authenticated and they have not run sync at least once. This is because we want to ensure the user has up to date items before they are allowed to modify anything.

Decrypting / Reading Items

You can decrypt synced items by calling the `user.decrypt` method on an item.

```
>>> print(user.decrypt(user.items['d3866137-f7f6-4b98-8b34-43cae416472e']))
'testing'
```

Encrypting / Creating Items

You can encrypt a new item by calling `user.encrypt` method with some content.

```
>>> item = user.encrypt('my content', 'my content type')
>>> print(item)
Item(uuid='3120ebf8-a6f7-4620-b99a-3e4a0233fcb1')
```

Note: However, this does not mean that the resulting item is synced or even setup to be synced.

The resulting item is currently only stored in memory. In order to add the item to the sync, you can call the `user.create` method with the resulting item.

```
>>> user.create(item)
```

Now the created item exists locally and will be synced up to the remote whenever `user.sync` is called again.

You can use the shortcut method `user.create_from` with an existing file to encrypt and create the item with one call.

```
>>> item = user.create_from('/path/to/existing/file')
```

This item **is** currently setup to be synced and will be the next time `user.sync` is called.

Deleting Items

Deleting items should be done through the method `user.delete`.

```
>>> user.delete(item)
```

This will toggle the `deleted` flag and setup the item to be synced to the remote the next time `user.sync` is called.

Updating Items

If you have changed the content of an item, you can setup the item for syncing using the `user.update` method.

```
>>> user.update(item)
```

This will cause the item to be re-synced the next time `user.sync` is called.

1.2 Contributing

When contributing to this repository, please first discuss the change you wish to make via an issue to the owners of this repository before submitting a pull request.

Important: We have an enforced style guide and a code of conduct. Please follow them in all your interactions with this project.

1.2.1 Style Guide

- We somewhat follow PEP8 and utilize Sphinx docstrings on **all** classes and functions.
- **We employee flake8 as our linter with exceptions to the following rules:**
  - D203
  - F401
  - E123
  - W503
  - E203
- Maximum line length is 88 characters.
- Maxiumum McCabe complexity is 12.
- Linting and test environments are configured via `tox.ini`.
- Imports are sorted using isort with multi-line output mode `3`.
- An `.editorconfig` file is included in this repository which dictates whitespace, indentation, and file encoding rules.
- Although `requirements.txt` and `requirements_dev.txt` do exist, Pipenv is utilized as the primary virtual environment and package manager for this project.
- We strictly utilize Semantic Versioning as our version specification.
- All Python source files are post-processed using ambv/black.

1.2.2 Issues

Issues should follow the included `ISSUE_TEMPLATE` found in `.github/ISSUE_TEMPLATE.md`.

- **Issues should contain the following sections:**
  - Expected Behavior
  - Current Behavior
  - Possible Solution
  - Steps to Reproduce (for bugs)
  - Context
  - Your Environment

These sections help the developers greatly by providing a large understanding of the context of the bug or requested feature without having to launch a full fleged discussion inside of the issue.

1.2.3 Pull Requests

Pull requests should follow the included `PULL_REQUEST_TEMPLATE` found in `.github/PULL_REQUEST_TEMPLATE.md`.

- Pull requests should always be from a **topic/feature/bugfix** (left side) branch. *Pull requests from master branches will not be merged.*
- Pull requests should not fail our requested style guidelines or linting checks.

1.2.4 Code of Conduct

Our code of conduct is taken directly from the Contributor Covenant since it directly hits all of the points we find necessary to address.

Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at stephen@bunn.io. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

CHAPTER 2

Project Reference

2.1 Standard File Package

The following is the automatically built documentation for the entire `standardfile` package.

2.1.1 user

class standardfile.user.**User**(*email*, *host='https://n3.standardnotes.org'*, *sync_parent='/home/docs/checkouts/readthedocs.org/user_builds/standardfile/checkouts/lat*

Bases: typing.Generic

Defines a Standard File user.

Defined by StandardFile at https://standardfile.org/#user.

session
The session to use for requests.

Returns The session to use for requests

Return type Session

mfa_required
Indicates if multifactor authentication is required for this user.

Returns True if mfa is required, otherwise False

Return type bool

mfa_key
The multifactor authentication url parameter key if mfa is required.

Returns The mfa parameter key if mfa is required, otherwise None

Return type str

authenticated
Indicates if the user is currently authenticated.

Returns True if the user is authenticated, otherwise False

Return type bool

sync_dir
Returns the sync directory for the user.

Returns The local sync directory

Return type Path

classmethod register(*email*, *password*, *host='https://n3.standardnotes.org'*, *cost=60000*, **args*, ***kwargs*)
Registers a new user in the Standard File server.

Parameters

- **email** (*str*) – The email to register
- **password** (*str*) – The password to register
- **host** (str) – The host to register to, defaults to constants.DEFAULT_HOST
- **host** – str, optional
- **cost** (int) – The password iteration cost, defaults to 60000
- **cost** – int, optional

Returns A new user instance

Return type T_User

classmethod login(*email*, *password*, *host='https://n3.standardnotes.org'*, *mfa=None*, **args*, ***kwargs*)
Shortcut to quickly create a new user instance given an email and password.

Parameters

- **email** (*str*) – The email of the user
- **password** (*str*) – The password of the user
- **host** (str) – The host to login to, defaults to constants.DEFAULT_HOST
- **host** – str, optional
- **mfa** (Optional[str]) – MFA code (if multi-factor authentication enabled), defaults to None
- **mfa** – str, optional

Returns A new user instance

Return type T_User

authenticate(*password*, *mfa=None*)
Logs the user into the standard file server.

Parameters

- **password** (*str*) – The password of the user
- **mfa** (Optional[str]) – The multifactor authentication code, defaults to None
- **mfa** – str, optional

Raises ***exceptions.MFARequired*** – If mfa is required but no mfa provided

Returns The newly authenticated user

Return type T_User

sync(*items=[]*, *full=False*)
Syncs the authenicated user's items.

Parameters

- **items** (*List[Item], optional*) – A list of items to sync to the remote, defaults to []
- **full** (*bool, optional*) – Indicates if sync should be full sync, defaults to False

Raises ***exceptions.AuthRequired*** – If the calling user is not authenticated

Returns The server response dictionary

Return type dict

encrypt(*content*, *content_type*)
Encrypts some content into a new item instance.

Parameters

- **content** (*str*) – The content to encrypt
- **content_type** (*str*) – Some kind of content type descriptor

Returns A new item instance

Return type *Item*

decrypt(*item*)
Decrypt a user's item.

Parameters **item** (Item) – The item to decrypt

Raises

- **ValueError** – If the item has no decryptable content
- ***exceptions.AuthRequired*** – If the calling user isn't authenticated yet
- ***exceptions.TamperDetected*** – When the local uuid doesn't match the item id

Returns The resulting content dictionary

Return type dict

create(*item*, *sync=False*)
Creates a new item on both the remote and local.

Parameters

- **item** (Item) – The item to create
- **sync** (bool) – True if sync should occur immediately, defaults to False
- **sync** – bool, optional

Raises **ValueError** –

- When item is already synced
- When item already exists locally

create_from(*filepath*, *content_type*, *sync=False*)
Creates a new item on both the remote and local from a file on the local.

Parameters

- **filepath** (*str*) – The filepath to add to the sync
- **content_type** (*str*) – The content type of the filepath
- **sync** (bool) – True if sync should occur immediately, defaults to False
- **sync** – bool, optional

Raises **ValueError** – When filepath does not exist

Returns The created item

Return type *Item*

delete(*item*, *sync=False*)
Deletes an item from the sync.

Parameters

- **item** (Item) – The item to delete
- **sync** (bool) – True if sync should occur immediately, defaults to False
- **sync** – bool, optional

Raises **ValueError** –

- When the item is not currently synced
- When the item does not exist locally

update(*item*, *sync=False*)
Updates the content of an existing item to the remote.

Parameters

- **item** (Item) – The item to update
- **sync** (bool) – True if sync should occur immediately, defaults to False
- **sync** – bool, optional

Raises **ValueError** –

- When the item is not currently synced
- When the item does not exist locally

2.1.2 item

class standardfile.item.**String**(*version*, *auth_hash*, *uuid*, *iv*, *cipher_text*)
Bases: typing.Generic

Defines a Standard File string.

classmethod is_valid(*string*)
Check if a given string is valid.

Parameters **string** (*str*) – The string the check

Returns True if valid, otherwise False

Return type bool

classmethod from_string(*string*)
Creates an instance from a string.

Parameters **string** (*str*) – The string to create an instance from

Returns An instance of `String`

Return type T_String

to_string()
Writes string out to a dictionary.

Returns The resulting string

Return type str

class standardfile.item.**Item**(*uuid*, *content*, *content_type*, *enc_item_key*, *deleted*, *created_at*, *updated_at*, *auth_hash=None*)
Bases: typing.Generic

Defines a Standard File item.

Defined by StandardFile at https://standardfile.org/#items.

classmethod from_dict(*item_dict*)
Creates an instance from a dictionary.

Parameters **item_dict** (*dict*) – The dictionary to create an item from

Returns An instance of `Item`

Return type T_Item

to_dict()
Writes item out to a dictionary.

Returns The resulting dictionary

Return type dict

2.1.3 cryptography

class standardfile.cryptography.**Cryptographer**
Bases: object

The cryptographer class namespace.

preferred_version = '002'

classmethod decrypt(*string*, *encryption_key*, *auth_key*)
Decrypts a string using a encryption and authentication key.

Parameters

- **string** (`String`) – The string to decrypt
- **encryption_key** (*bytes*) – The encryption key to use
- **auth_key** (*bytes*) – The authentication key to use

Raises **ValueError** – If the string version is not supported

Returns The decrypted string

Return type str

classmethod encrypt(*content*, *uuid*, *encryption_key*, *auth_key*)
Encrypts a string using a encryption and authentication key.

Parameters

- **content** (*str*) – The content to encrypt

- **uuid** (*str*) – The desired uuid string of the new content
- **encryption_key** (*bytes*) – The encryption key to use
- **auth_key** (*bytes*) – The authentication key to use

Raises **ValueError** – If the preferred version is not supported

Returns The resulting string instance

Return type *String*

2.1.4 exceptions

exception standardfile.exceptions.**StandardFileException**(*message*, *data={}*)
Bases: Exception

The parent exception of all custom Standard File exceptions.

exception standardfile.exceptions.**AuthException**(*message*, *data={}*)
Bases: *standardfile.exceptions.StandardFileException*

The exception namespace for all authentication based exceptions.

exception standardfile.exceptions.**AuthRequired**(*message*, *data={}*)
Bases: *standardfile.exceptions.AuthException*

Raised when authentication is required but not provided.

exception standardfile.exceptions.**AuthInvalid**(*message*, *data={}*)
Bases: *standardfile.exceptions.AuthException*

Raised when authentication is invalid.

exception standardfile.exceptions.**MFARequired**(*message*, *data={}*)
Bases: *standardfile.exceptions.AuthException*

Raised when multifactor authentication is required but not provided.

exception standardfile.exceptions.**MFAInvalid**(*message*, *data={}*)
Bases: *standardfile.exceptions.AuthException*

Raised when the provided multifactor authentication is invalid.

exception standardfile.exceptions.**TamperException**(*message*, *data={}*)
Bases: *standardfile.exceptions.StandardFileException*

The exception namespace for all tamper based exceptions.

exception standardfile.exceptions.**TamperDetected**(*message*, *data={}*)
Bases: *standardfile.exceptions.TamperException*

Raised when string tampering is detected.

Python Module Index

S

standardfile, 9
standardfile.cryptography, 13
standardfile.exceptions, 14
standardfile.item, 12
standardfile.user, 9

Index

A

authenticate() (standardfile.user.User method), 10
authenticated (standardfile.user.User attribute), 9
AuthException, 14
AuthInvalid, 14
AuthRequired, 14

C

create() (standardfile.user.User method), 11
create_from() (standardfile.user.User method), 11
Cryptographer (class in standardfile.cryptography), 13

D

decrypt() (standardfile.cryptography.Cryptographer class method), 13
decrypt() (standardfile.user.User method), 11
delete() (standardfile.user.User method), 12

E

encrypt() (standardfile.cryptography.Cryptographer class method), 13
encrypt() (standardfile.user.User method), 11

F

from_dict() (standardfile.item.Item class method), 13
from_string() (standardfile.item.String class method), 12

I

is_valid() (standardfile.item.String class method), 12
Item (class in standardfile.item), 13

L

login() (standardfile.user.User class method), 10

M

mfa_key (standardfile.user.User attribute), 9
mfa_required (standardfile.user.User attribute), 9
MFAInvalid, 14
MFARequired, 14

P

preferred_version (standardfile.cryptography.Cryptographer attribute), 13

R

register() (standardfile.user.User class method), 10

S

session (standardfile.user.User attribute), 9
standardfile (module), 9
standardfile.cryptography (module), 13
standardfile.exceptions (module), 14
standardfile.item (module), 12
standardfile.user (module), 9
StandardFileException, 14
String (class in standardfile.item), 12
sync() (standardfile.user.User method), 11
sync_dir (standardfile.user.User attribute), 10

T

TamperDetected, 14
TamperException, 14
to_dict() (standardfile.item.Item method), 13
to_string() (standardfile.item.String method), 13

U

update() (standardfile.user.User method), 12
User (class in standardfile.user), 9