

---

# Python

unknown

Mar 24, 2022



# CONTENTS

<b>1</b>	<b>History</b>	<b>1</b>
<b>2</b>	<b>Indices and tables</b>	<b>7</b>
	<b>Python Module Index</b>	<b>9</b>
	<b>Index</b>	<b>11</b>



## HISTORY

```
class keyrings.alt.file.Encrypted
```

Bases: object

PyCryptodome-backed Encryption support

```
block_size = 32
```

```
scheme = '[PBKDF2] AES256.CFB'
```

```
version = '1.0'
```

```
class keyrings.alt.file.EncryptedKeyring
```

Bases: *keyrings.alt.file.Encrypted*, *keyrings.alt.file\_base.Keyring*

PyCryptodome File Keyring

```
decrypt(password_encrypted, assoc=None)
```

Given a password encrypted by a previous call to *encrypt*, and *assoc* (byte string, optional), return the original byte string.

*assoc* provides associated data (typically: service and username)

```
encrypt(password, assoc=None)
```

Given a password (byte string) and *assoc* (byte string, optional), return an encrypted byte string.

*assoc* provides associated data (typically: service and username)

```
filename = 'crypted_pass.cfg'
```

```
keyring_key
```

Much like the property builtin, but only implements `__get__`, making it a non-data property, and can be subsequently reset.

See <http://users.rcn.com/python/download/Descriptor.htm> for more information.

```
>>> class X:
...     @NonDataProperty
...     def foo(self):
...         return 3
>>> x = X()
>>> x.foo
3
>>> x.foo = 4
>>> x.foo
4
```

```
priority = 0.6
```

```
pw_prefix = b'pw:'

class keyrings.alt.file.PlaintextKeyring
    Bases: keyrings.alt.file_base.Keyring

    Simple File Keyring with no encryption

    decrypt(password_encrypted, assoc=None)
        Directly return encrypted password, ignore associated data.

    encrypt(password, assoc=None)
        Directly return the password itself, ignore associated data.

    filename = 'keyring_pass.cfg'

    priority = 0.5
        Applicable for all platforms, but not recommended

    scheme = 'no encryption'

    version = '1.0'

class keyrings.alt.Gnome.Keyring
    Bases: keyring.backend.KeyringBackend

    Gnome Keyring

    KEYRING_NAME = None
        Name of the keyring in which to store the passwords. Use None for the default keyring.

    delete_password(service, username)
        Delete the password for the username of the service.

    get_password(service, username)
        Get password of the username for the service

    property keyring_name

    property priority
        Each backend class must supply a priority, a number (float or integer) indicating the priority of the backend
        relative to all other backends. The priority need not be static – it may (and should) vary based attributes of
        the environment in which is runs (platform, available packages, etc.).

        A higher number indicates a higher priority. The priority should raise a RuntimeError with a message
        indicating the underlying cause if the backend is not suitable for the current environment.

        As a rule of thumb, a priority between zero but less than one is suitable, but a priority of one or greater is
        recommended.

    set_password(service, username, password)
        Set password for the username of the service

class keyrings.alt.Google.DocsKeyring(credential, source, crypter, collection=None, client=None,
    can_create=True, input_getter=<built-in function input>)
    Bases: keyring.backend.KeyringBackend

    Backend that stores keyring on Google Docs. Note that login and any other initialisation is deferred until it is
    actually required to allow this keyring class to be added to the global _all_keyring list.

    CONFLICT = -1

    FAIL = 0

    OK = 1

    property client
```

**property collection****delete\_password**(*service, username*)

Delete the password for the username of the service.

If the backend cannot store passwords, raise `NotImplementedError`.**get\_password**(*service, username*)

Get password of the username for the service

**keyring\_title** = 'GoogleKeyring'**property priority**

Each backend class must supply a priority, a number (float or integer) indicating the priority of the backend relative to all other backends. The priority need not be static – it may (and should) vary based attributes of the environment in which is runs (platform, available packages, etc.).

A higher number indicates a higher priority. The priority should raise a `RuntimeError` with a message indicating the underlying cause if the backend is not suitable for the current environment.

As a rule of thumb, a priority between zero but less than one is suitable, but a priority of one or greater is recommended.

**set\_password**(*service, username, password*)

Set password for the username of the service

**class** `keyrings.alt.Google.EnvironCredential`Bases: `keyring.credentials.EnvironCredential`

Retrieve credentials from specifically named environment variables

**class** `keyrings.alt.Google.KeyczarDocsKeyring`Bases: `keyrings.alt.Google.DocsKeyring`

Google Docs keyring using keyczar initialized from environment variables

**supported**()

Return if this keyring supports current environment: -1: not applicable

0: suitable 1: recommended

**class** `keyrings.alt.keyczar.BaseCrypter`Bases: `keyring.backend.Crypter`

Base Keyczar keyset encryption and decryption. The keyset initialisation is deferred until required.

**property crypter**

The actual keyczar crypter

**decrypt**(*value*)

Decrypt the value.

**encrypt**(*value*)

Encrypt the value.

**abstract property encrypting\_keyset\_location**Location for the encrypting keyset. Use `None` to indicate that the main keyset is not encrypted**abstract property keyset\_location**

Location for the main keyset that may be encrypted or not

**class** `keyrings.alt.keyczar.Crypter`(*keyset\_location, encrypting\_keyset\_location=None*)Bases: `keyrings.alt.keyczar.BaseCrypter`

A Keyczar crypter using locations specified in the constructor

**property encrypting\_keyset\_location**

Location for the encrypting keyset. Use None to indicate that the main keyset is not encrypted

**property keyset\_location**

Location for the main keyset that may be encrypted or not

**class** keyrings.alt.keyczar.**EnvironCrypter**

Bases: [keyrings.alt.keyczar.BaseCrypter](#)

A Keyczar crypter using locations specified by environment vars

**ENC\_KEYSET\_ENV\_VAR** = 'KEYRING\_KEYCZAR\_ENCRYPTING\_LOCATION'

**KEYSET\_ENV\_VAR** = 'KEYRING\_KEYCZAR\_ENCRYPTED\_LOCATION'

**property encrypting\_keyset\_location**

Location for the encrypting keyset. Use None to indicate that the main keyset is not encrypted

**property keyset\_location**

Location for the main keyset that may be encrypted or not

**keyrings.alt.keyczar.has\_keyczar()****class** keyrings.alt.multi.**MultipartKeyringWrapper**(keyring, max\_password\_size=512)

Bases: [keyring.backend.KeyringBackend](#)

A wrapper around an existing keyring that breaks the password into smaller parts to handle implementations that have limits on the maximum length of passwords i.e. Windows Vault

**delete\_password**(service, username)

Delete the password for the username of the service.

If the backend cannot store passwords, raise `NotImplementedError`.

**get\_password**(service, username)

Get password of the username for the service

**priority** = 0

**set\_password**(service, username, password)

Set password for the username of the service

**class** keyrings.alt.pyfs.**BasicKeyring**(crypter, filename=None, can\_create=True, cache\_timeout=None)

Bases: [keyring.backend.KeyringBackend](#)

BasicKeyring is a Pyfilesystem-based implementation of keyring.

It stores the password directly in the file, and supports encryption and decryption. The encrypted password is stored in base64 format. Being based on Pyfilesystem the file can be local or network-based and served by any of the filesystems supported by Pyfilesystem including Amazon S3, FTP, WebDAV, memory and more.

**property config**

load the passwords from the config file

**decrypt**(password\_encrypted)

Decrypt the password.

**delete\_password**(service, username)

Delete the password for the username of the service.

If the backend cannot store passwords, raise `NotImplementedError`.

**encrypt**(password)

Encrypt the password.



**file\_path**

Much like the property builtin, but only implements `__get__`, making it a non-data property, and can be subsequently reset.

See <http://users.rcn.com/python/download/Descriptor.htm> for more information.

```
>>> class X:
...     @NonDataProperty
...     def foo(self):
...         return 3
>>> x = X()
>>> x.foo
3
>>> x.foo = 4
>>> x.foo
4
```

**property filename**

The filename used to store the passwords.

**get\_password(service, username)**

Read the password from the file.

**property priority**

Each backend class must supply a priority, a number (float or integer) indicating the priority of the backend relative to all other backends. The priority need not be static – it may (and should) vary based attributes of the environment in which is runs (platform, available packages, etc.).

A higher number indicates a higher priority. The priority should raise a `RuntimeError` with a message indicating the underlying cause if the backend is not suitable for the current environment.

As a rule of thumb, a priority between zero but less than one is suitable, but a priority of one or greater is recommended.

**set\_password(service, username, password)**

Write the password in the file.

```
class keyrings.alt.pyfs.EncryptedKeyring(crypter, filename=None, can_create=True,
                                         cache_timeout=None)
```

Bases: `keyrings.alt.pyfs.BasicKeyring`

Encrypted Pyfilesystem Keyring

```
class keyrings.alt.pyfs.KeyczarKeyring
```

Bases: `keyrings.alt.pyfs.EncryptedKeyring`

Encrypted Pyfilesystem Keyring using Keyczar keysets specified in environment vars

```
class keyrings.alt.pyfs.PlaintextKeyring(filename=None, can_create=True, cache_timeout=None)
```

Bases: `keyrings.alt.pyfs.BasicKeyring`

Unencrypted Pyfilesystem Keyring

```
keyrings.alt.pyfs.has_pyfs()
```

Does this environment have pyfs 1.x installed? Should return False even when Mercurial's Demand Import allowed import of fs.\*.

```
class keyrings.alt.Windows.EncryptedKeyring
```

Bases: `keyrings.alt.file_base.Keyring`

A File-based keyring secured by Windows Crypto API.

**decrypt**(*password\_encrypted*, *assoc=None*)  
Decrypt the password using the CryptAPI.

**encrypt**(*password*, *assoc=None*)  
Encrypt the password using the CryptAPI.

**filename** = 'wincrypto\_pass.cfg'

**property priority**  
Preferred over file.EncryptedKeyring but not other, more sophisticated Windows backends.

**version** = '1.0'

**class** keyrings.alt.Windows.**RegistryKeyring**

Bases: keyring.backend.KeyringBackend

RegistryKeyring is a keyring which use Windows CryptAPI to encrypt the user's passwords and store them under registry keys

**delete\_password**(*service*, *username*)  
Delete the password for the username of the service.

**get\_password**(*service*, *username*)  
Get password of the username for the service

**property priority**  
Preferred on Windows when pywin32 isn't installed

**set\_password**(*service*, *username*, *password*)  
Write the password to the registry

keyrings.alt.Windows.**has\_wincrypto**()

Does this environment have wincrypto? Should return False even when Mercurial's Demand Import allowed import of \_win\_crypto, so accesses an attribute of the module.

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### k

- `keyrings.alt.file`, 1
- `keyrings.alt.Gnome`, 2
- `keyrings.alt.Google`, 2
- `keyrings.alt.keyczar`, 3
- `keyrings.alt.multi`, 4
- `keyrings.alt.pyfs`, 4
- `keyrings.alt.Windows`, 5



## B

**BaseCrypter** (class in *keyrings.alt.keyczar*), 3  
**BasicKeyring** (class in *keyrings.alt.pyfs*), 4  
**block\_size** (*keyrings.alt.file.Encrypted* attribute), 1

## C

**client** (*keyrings.alt.Google.DocsKeyring* property), 2  
**collection** (*keyrings.alt.Google.DocsKeyring* property), 2  
**config** (*keyrings.alt.pyfs.BasicKeyring* property), 4  
**CONFLICT** (*keyrings.alt.Google.DocsKeyring* attribute), 2  
**Crypter** (class in *keyrings.alt.keyczar*), 3  
**crypter** (*keyrings.alt.keyczar.BaseCrypter* property), 3

## D

**decrypt()** (*keyrings.alt.file.EncryptedKeyring* method), 1  
**decrypt()** (*keyrings.alt.file.PlaintextKeyring* method), 2  
**decrypt()** (*keyrings.alt.keyczar.BaseCrypter* method), 3  
**decrypt()** (*keyrings.alt.pyfs.BasicKeyring* method), 4  
**decrypt()** (*keyrings.alt.Windows.EncryptedKeyring* method), 5  
**delete\_password()** (*keyrings.alt.Gnome.Keyring* method), 2  
**delete\_password()** (*keyrings.alt.Google.DocsKeyring* method), 3  
**delete\_password()** (*keyrings.alt.multi.MultipartKeyringWrapper* method), 4  
**delete\_password()** (*keyrings.alt.pyfs.BasicKeyring* method), 4  
**delete\_password()** (*keyrings.alt.Windows.RegistryKeyring* method), 6  
**DocsKeyring** (class in *keyrings.alt.Google*), 2

## E

**ENC\_KEYSET\_ENV\_VAR** (*keyrings.alt.keyczar.EnvronCrypter* attribute), 4  
**encrypt()** (*keyrings.alt.file.EncryptedKeyring* method), 1  
**encrypt()** (*keyrings.alt.file.PlaintextKeyring* method), 2  
**encrypt()** (*keyrings.alt.keyczar.BaseCrypter* method), 3  
**encrypt()** (*keyrings.alt.pyfs.BasicKeyring* method), 4

**encrypt()** (*keyrings.alt.Windows.EncryptedKeyring* method), 6  
**Encrypted** (class in *keyrings.alt.file*), 1  
**EncryptedKeyring** (class in *keyrings.alt.file*), 1  
**EncryptedKeyring** (class in *keyrings.alt.pyfs*), 5  
**EncryptedKeyring** (class in *keyrings.alt.Windows*), 5  
**encrypting\_keyset\_location** (*keyrings.alt.keyczar.BaseCrypter* property), 3  
**encrypting\_keyset\_location** (*keyrings.alt.keyczar.Crypter* property), 3  
**encrypting\_keyset\_location** (*keyrings.alt.keyczar.EnvronCrypter* property), 4  
**EnvironCredential** (class in *keyrings.alt.Google*), 3  
**EnvironCrypter** (class in *keyrings.alt.keyczar*), 4

## F

**FAIL** (*keyrings.alt.Google.DocsKeyring* attribute), 2  
**file\_path** (*keyrings.alt.pyfs.BasicKeyring* attribute), 4  
**filename** (*keyrings.alt.file.EncryptedKeyring* attribute), 1  
**filename** (*keyrings.alt.file.PlaintextKeyring* attribute), 2  
**filename** (*keyrings.alt.pyfs.BasicKeyring* property), 5  
**filename** (*keyrings.alt.Windows.EncryptedKeyring* attribute), 6

## G

**get\_password()** (*keyrings.alt.Gnome.Keyring* method), 2  
**get\_password()** (*keyrings.alt.Google.DocsKeyring* method), 3  
**get\_password()** (*keyrings.alt.multi.MultipartKeyringWrapper* method), 4  
**get\_password()** (*keyrings.alt.pyfs.BasicKeyring* method), 5  
**get\_password()** (*keyrings.alt.Windows.RegistryKeyring* method), 6

## H

**has\_keyczar()** (in module *keyrings.alt.keyczar*), 4  
**has\_pyfs()** (in module *keyrings.alt.pyfs*), 5  
**has\_wincrypto()** (in module *keyrings.alt.Windows*), 6

## K

KeyczarDocsKeyring (class in *keyrings.alt.Google*), 3  
KeyczarKeyring (class in *keyrings.alt.pyfs*), 5  
Keyring (class in *keyrings.alt.Gnome*), 2  
keyring\_key (*keyrings.alt.file.EncryptedKeyring* attribute), 1  
KEYRING\_NAME (*keyrings.alt.Gnome.Keyring* attribute), 2  
keyring\_name (*keyrings.alt.Gnome.Keyring* property), 2  
keyring\_title (*keyrings.alt.Google.DocsKeyring* attribute), 3  
*keyrings.alt.file* module, 1  
*keyrings.alt.Gnome* module, 2  
*keyrings.alt.Google* module, 2  
*keyrings.alt.keyczar* module, 3  
*keyrings.alt.multi* module, 4  
*keyrings.alt.pyfs* module, 4  
*keyrings.alt.Windows* module, 5  
KEYSET\_ENV\_VAR (*keyrings.alt.keyczar.EnvionCrypter* attribute), 4  
keyset\_location (*keyrings.alt.keyczar.BaseCrypter* property), 3  
keyset\_location (*keyrings.alt.keyczar.Crypter* property), 4  
keyset\_location (*keyrings.alt.keyczar.EnvionCrypter* property), 4

## M

module  
    *keyrings.alt.file*, 1  
    *keyrings.alt.Gnome*, 2  
    *keyrings.alt.Google*, 2  
    *keyrings.alt.keyczar*, 3  
    *keyrings.alt.multi*, 4  
    *keyrings.alt.pyfs*, 4  
    *keyrings.alt.Windows*, 5  
MultipartKeyringWrapper (class in *keyrings.alt.multi*), 4

## O

OK (*keyrings.alt.Google.DocsKeyring* attribute), 2

## P

PlaintextKeyring (class in *keyrings.alt.file*), 2  
PlaintextKeyring (class in *keyrings.alt.pyfs*), 5  
priority (*keyrings.alt.file.EncryptedKeyring* attribute), 1

priority (*keyrings.alt.file.PlaintextKeyring* attribute), 2  
priority (*keyrings.alt.Gnome.Keyring* property), 2  
priority (*keyrings.alt.Google.DocsKeyring* property), 3  
priority (*keyrings.alt.multi.MultipartKeyringWrapper* attribute), 4  
priority (*keyrings.alt.pyfs.BasicKeyring* property), 5  
priority (*keyrings.alt.Windows.EncryptedKeyring* property), 6  
priority (*keyrings.alt.Windows.RegistryKeyring* property), 6  
pw\_prefix (*keyrings.alt.file.EncryptedKeyring* attribute), 1

## R

RegistryKeyring (class in *keyrings.alt.Windows*), 6

## S

scheme (*keyrings.alt.file.Encrypted* attribute), 1  
scheme (*keyrings.alt.file.PlaintextKeyring* attribute), 2  
set\_password() (*keyrings.alt.Gnome.Keyring* method), 2  
set\_password() (*keyrings.alt.Google.DocsKeyring* method), 3  
set\_password() (*keyrings.alt.multi.MultipartKeyringWrapper* method), 4  
set\_password() (*keyrings.alt.pyfs.BasicKeyring* method), 5  
set\_password() (*keyrings.alt.Windows.RegistryKeyring* method), 6  
supported() (*keyrings.alt.Google.KeyczarDocsKeyring* method), 3

## V

version (*keyrings.alt.file.Encrypted* attribute), 1  
version (*keyrings.alt.file.PlaintextKeyring* attribute), 2  
version (*keyrings.alt.Windows.EncryptedKeyring* attribute), 6