python-sdbus-secrets

igo95862

Sep 25, 2021

CONTENTS:

This package contains python-sdbus binds for Freedesktop secrets standard.

CHAPTER

ONE

FREEDESKTOP SECRETS TUTORIAL

This tutorial will guide you through the basic steps on how to use Freedesktop Secrets API using the python-sdbus binds.

Note: This tutorial will use blocking API for simplicity. Async API can be easily used instead.

1.1 Creating a new session

To interact with the secrets API a process needs to acquire a new session.

Secrets API supports an in-transit encryption but its not required. Most of secrets implementations should support plain mode without encryption. This is the simplest way to use secrets API.

```
secrets_service = SecretService()

session_algorithm = 'plain'  # Plain mode, no encryption
session_input = ('s', '')  # Variant of an empty string

# With plain algorithm we only need session path
# output of algorithm negotiation can be ignored (assigned to _)
_, my_session_path = secrets_service.open_session(
    session_algorithm,
    session_input,
)
```

Session is identified by an object path. In this case the session path is stored in the `my_session_path` variable.

Session can be closed manually using *SecretSessionInterface.close()* method or automatically when process that acquired session disconnects from the bus.

1.2 Acquiring default collection

Unless you need to store multiple secrets it is better to find a default collection.

This collection should always be present.

```
default_collection_path = secrets_service.read_alias('default')

default_collection = SecretCollection(default_collection_path)
```

1.3 Creating secrets

SecretCollectionInterface.create_item() should be used to create new items.

The secret can be created with properties to uniquely identify it. Each property name must be prefixed with `org.freedesktop.Secret.Item.` string. For example, secret attributes would be `org.freedesktop.Secret.Item.Attributes`. See example.

The secret data itself is a tuple of session path, encryption parameters bytes (empty in case of plain mode), bytes of value and content type string. (for example, `text/plain; charset=utf8`)

Last argument is a boolean whether or not to replace existing secret with same attributes.

```
secret_properties_dict = {
    'org.freedesktop.Secret.Item.Label': ('s', 'MyItem'),
    'org.freedesktop.Secret.Item.Type': ('s', 'Test'),
    'org.freedesktop.Secret.Item.Attributes': ('a{ss}', {
        "Attribute1": "Value1",
        "Attribute2": "Value2",
    })
}

new_secret_path, prompt = default_collection.create_item(
    secret_properties_dict,
    (
        my_session_path,  # session path
        b'',  # encryption parameters, empty in plain mode
        b'my secret',  # secret value it self
        'text/plain; charset=utf8',  # content type
    ),
    False,  # do not replace secret with same attributes
)
```

1.4 Searching secrets

After getting a collection you can either search the items using *SecretCollectionInterface.search_items()* or iterate over *SecretCollectionInterface.items()* property and examine each secret individually.

Each secret has a dictionary of attributes which can be used to uniquely identify a secret.

```
found_secrets_paths = default_collection.search_items(
    {
        "Attribute1": "Value1",
        "Attribute2": "Value2",
    }
)
```

1.5 Getting secrets

After finding the secret path in order to get the secret you should use the *SecretItemInterface.get_secret()* method to get secret data.

Secret data contains tuple of session path, encryption parameters bytes (empty in case of plain mode), secret value bytes and content type string.

```
secret = SecretItem(new_secret_path)

session_path, params, value, content_type = secret.get_secret(my_session_path)
```

Note: See secrets specification for more in depth look.

CHAPTER

TWO

SECRETS OBJECTS

class sdbus_async.secrets.**SecretService**(*bus=None*)
Secret service main object.

Implements *SecretServiceInterface*

Bus name and object path is predetermined at `org.freedesktop.secrets` and `/org/freedesktop/secrets` respectively.

Parameters **bus** (*SdBus*) – Use specific bus or session bus by default.

Return type None

class sdbus_async.secrets.**SecretCollection**(*collection_path*, *bus=None*)
Secrets collection.

Implements *SecretCollectionInterface*

Bus name is predetermined at `org.freedesktop.secrets`

Parameters

- **collection_path** (*str*) – Object path to collection.
- **bus** (*SdBus*) – Use specific bus or session bus by default.

Return type None

class sdbus_async.secrets.**SecretItem**(*item_path*, *bus=None*)
Secrets item.

Implements *SecretItemInterface*

Bus name is predetermined at `org.freedesktop.secrets`

Parameters

- **item_path** (*str*) – Object path to item.
- **bus** (*SdBus*) – Use specific bus or session bus by default.

Return type None

class sdbus_async.secrets.**SecretPrompt**(*prompt_path*, *bus=None*)
Secrets prompt.

Implements *SecretPromptInterface*

Bus name is predetermined at `org.freedesktop.secrets`

Parameters

- **prompt_path** (*str*) – Object path to prompt.

- **bus** (*SdBus*) – Use specific bus or session bus by default.

Return type None

class sdbus_async.secrets.**SecretSession**(*session_path*, *bus=None*)

Secrets session.

Implements *SecretSessionInterface*

Bus name is predetermined at `org.freedesktop.secrets`

Parameters

- **session_path** (*str*) – Object path to session.
- **bus** (*SdBus*) – Use specific bus or session bus by default.

Return type None

CHAPTER

THREE

SECRETS INTERFACES

All secrets D-Bus interfaces.

class sdbus_async.secrets.**SecretServiceInterface**

Secrets daemon interface.

Used to create new sessions and etc...

Return type None

async open_session(*algorithm*, *input*)

D-Bus Method

Create new session.

Parameters

- **algorithm** (*str*) – Session algorithm. The `plain` algorithm type with no encryption is always supported.
- **input** (*Tuple[str,Any]*) – Input arguments for the algorithm.

Returns Tuple of output of the algorithm negotiation and object path of the new session.

Return type Tuple[Tuple[str,Any],str]

async create_collection(*properties*, *alias*)

D-Bus Method

Create a new collection with the specified properties.

If new collection object path is `/` promting is necessary.

If the returned prompt object path is `/` no prompt is needed.

Parameters

- **properties** (*Dict[str,Tuple[str,Any]]*) – Dict of variants with properties of the new collection.
- **alias** (*str*) – Set this to an empty string if the new collection should not be associated with a well known alias. (such as `default`)

Returns Tuple of object path of new collection and possible object path of prompt object.

Return type Tuple[str,str]

async search_items(*attributes*)

D-Bus Method

Find items in any collection.

Parameters **attributes** (*Dict[str,str]*) – Attributes that should match.

Returns Two arrays of matched object paths. First arrays contains unlocked items and second locked ones.

Return type Tuple[List[str],List[str]]

`async unlock`(*objects*)
D-Bus Method

Unlock the specified objects.

Parameters **`objects`** (*`List[str]`*) – List of object paths to unlock

Returns List of objects unlocked without prompt and a path to prompt object. (has a value of `/` if no prompt needed)

Return type Tuple[List[str], str]

`async lock`(*objects*)
D-Bus Method

Lock items.

Parameters **`objects`** (*`List[str]`*) – List of object paths to lock

Returns List of objects locked without prompt and a path to prompt object. (has a value of `/` if no prompt needed)

Return type Tuple[List[str], str]

`async get_secrets`(*items*, *session*)
D-Bus Method

Retrieve multiple secrets from different items.

Parameters

- **`items`** (*`List[str]`*) – List of object paths to items.
- **`session`** (*`str`*) – Object path of current session.

Returns Dictionary with keys as requested object paths and values as secret items data.

Return type Dict[str,Tuple[str,bytes,bytes,str]]

`async read_alias`(*name*)
D-Bus Method

Get the collection with the given alias.

Parameters **`name`** (*`str`*) – An alias, such as `default`.

Retuns Object path to collection or `/` if no such alias exists.

Return type str

`async set_alias`(*name*, *collection*)
D-Bus Method

Setup a collection alias.

Parameters

- **`name`** (*`str`*) – The alias to use.
- **`collection`** (*`str`*) – Object path to collection to apply alias to.

Return type None

`collections: List[str]`
D-Bus property

Object paths of all collections.

`collection_created: str`
D-Bus signal

Signal when collection has been created.

Signal data is an object path to new collection.

`collection_deleted: str`
D-Bus signal

Signal when collection was deleted.

Signal data is an object path of removed collection.

`collection_changed: str`
D-Bus signal

Signal when a collection was modified.

Signal data is the modified collection object path.

`class sdbus_async.secrets.SecretCollectionInterface`
Collection of items containing secrets.

Return type None

`async delete()`
D-Bus Method

Delete this collection.

Returns Object path of the prompt or `/` if no prompt is needed.

Return type str

`async search_items(attributes)`
D-Bus Method

Search for items in this collection.

Parameters **attributes** (*Dict[str,str]*) – Attributes that should match.

Returns List of matched items object paths.

Return type List[str]

`async create_item(properties, secret, replace)`
D-Bus Method

Create new item.

Parameters

- **properties** (*Dict[str,Tuple[str,Any]]*) – Set properties of the new item. The keys are names of properties with prefixed with `org.freedesktop.Secret.Item.`. For example, `label` property will have a `org.freedesktop.Secret.Item.Label` key.
- **secret** (*Tuple[str,bytes,bytes,str]*) – Secret data. Secret data contains tuple of session path, encryption parameters bytes (empty in case of plain mode), secret value bytes and content type string.
- **replace** (*bool*) – Replace existing item with same attributes.

Returns Object path of new item or / if prompt needed and object path of prompt or / if prompt is not needed.

Return type Tuple[str,str]

`items: List[str]`
D-Bus property

List of object paths of items in this colletion.

`label: str`
D-Bus property

Display name of this collection.

`locked: bool`
D-Bus property

Whether the collection is locked or not.

`created: int`
D-Bus property

Unix time of creation.

`modified: int`
D-Bus property

Unix time of last modified.

`item_created: str`
D-Bus signal

Signal when new item was created.

Signal data is object path of new item.

`item_deleted: str`
D-Bus signal

Signal when item was deleted.

Signal data is object path of deleted item.

`item_changed: str`
D-Bus signal

Signal when an item was changed.

Signal data is object path of changed item.

`class sdbus_async.secrets.SecretItemInterface`
Item containing a secret.

Return type None

`async delete()`
D-Bus Method

Delete this item.

Returns Path to prompt or / if no prompt necessary.

Return type str

async get_secret(*session*)
D-Bus Method

Get secret of this item.

Returns Secret data. Secret data contains tuple of session path, encryption parameters bytes (empty in case of plain mode), secret value bytes and content type string.

Return type Tuple[str,bytes,bytes,str]

Parameters **session** (*str*) –

async set_secret(*secret*)
D-Bus Method

Set the secret for this item.

Parameters **secret** (*Tuple[str,bytes,bytes,str]*) – Secret data. Secret data contains tuple of session path, encryption parameters bytes (empty in case of plain mode), secret value bytes and content type string.

Return type None

locked: bool
D-Bus property

Is secret locked?

attributes: Dict[str, str]
D-Bus property

Item attributes.

label: str
D-Bus property

Item display name.

created: int
D-Bus property

Unix time of creation.

modified: int
D-Bus property

Unix time of last modified.

class sdbus_async.secrets.**SecretPromptInterface**
Prompt necessary to complete and operation.

Return type None

async prompt(*window_id*)
D-Bus Method

Preform a prompt.

Parameters **window_id** (*str*) – Platform specific window handle to use for showing the prompt.

Return type None

async dismiss()
D-Bus Method

Dismiss the prompts.

Return type None

`completed: Tuple[bool, Tuple[str, Any]]`
D-Bus signal

Signal when prompt is completed.

Signal data is:

- Boolean whether the prompt was dismissed or not.
- Possibly empty, operation specific result.

`class sdbus_async.secrets.SecretSessionInterface`
Session state between client and service.

Return type None

`async close()`
D-Bus Method

Close this session.

Return type None

INDEX

S

U