Craft Store

Release 2.2.1

Canonical Ltd.

Sep 23, 2022

GETTING STARTED

CHAPTER

ONE

TUTORIALS

1.1 Login to the Snap Store

1.1.1 Prerequisites

- Python 3.8 or 3.9
- a clean virtual environment setup
- a text editor
- a developer account on https://snapcraft.io

1.1.2 Setup

Enable the virtual environment and then install Craft Store by running:

```
$ pip install craft-store
```

1.1.3 Code

Write following into a a text editor and save it as `snap_store_login.py`:

```
#!/usr/bin/env python
from craft_store import StoreClient, endpoints

store_client = StoreClient(
    base_url="https://dashboard.snapcraft.io",
    storage_base_url="https://upload.apps.staging.ubuntu.com",
    endpoints=endpoints.SNAP_STORE,
    user_agent="Craft Store Tutorial Agent",
    application_name="cart-store-tutorial"
)

store_client.login(
    permissions=["package_access"],
    description="tutorial-client-login",
    ttl=1000
)
```

1.1.4 Run

Run the saved python module to login:

```
$ python snap_store_login.py
```

1.2 Login to the Snap Store using Ubuntu One

At the end of this tutorial you will have successfully written a script that can log you into the Snap Store using Ubuntu One (https://login.ubuntu.com) and have those credentials stored for the combination of Dashboard endpoint (https://dashboard.snapcraft.io) and application name (`ubuntu1-dashboard`).

1.2.1 Prerequisites

- Python 3.8 or 3.9
- a clean virtual environment setup
- a text editor
- a developer account on https://snapcraft.io

1.2.2 Setup

Enable the virtual environment and then install Craft Store by running:

```
$ pip install craft-store click
```

1.2.3 Code

Write following into a a text editor and save it as `snap_store_login_ubuntu_one.py`:

```
import click

from craft_store import *

c = UbuntuOneStoreClient(
    base_url="https://dashboard.snapcraft.io",
    storage_base_url="https://upload.apps.staging.ubuntu.com",
    auth_url="https://login.ubuntu.com",
    endpoints=endpoints.U1_SNAP_STORE,
    application_name="ubuntu1-dashboard",
    user_agent="test",
)

email = click.prompt("Email")
password = click.prompt("Password", hide_input=True)

try:
    c.login(
```

(continues on next page)

(continued from previous page)

```
        permissions=[
            "package_access",
            "package_manage",
            "package_metrics",
            "package_push",
            "package_register",
            "package_release",
            "package_update",
        ],
        description="foo",
        ttl=1800,
        email=email,
        password=password,
    )
except errors.StoreServerError as server_error:
    if "twofactor-required" in server_error.error_list:
        otp = click.prompt("OTP")
        c.login(
            permissions=[
                "package_access",
                "package_manage",
                "package_metrics",
                "package_push",
                "package_register",
                "package_release",
                "package_update",
            ],
            description="foo",
            ttl=1800,
            email=email,
            password=password,
            otp=otp,
        )
```

1.2.4 Run

Run the saved python module to login:

```
$ python snap_store_login_ubuntu_one.py
```

1.3 Get Account email and id from the Snap Store

1.3.1 Prerequisites

- Completed *Login to the Snap Store*
- Shelled into the virtual environment created in *Login to the Snap Store*

1.3.2 Code

Write following into a a text editor and save it as `snap_store_whoami.py`:

```
#!/usr/bin/env python
from craft_store import StoreClient, endpoints

store_client = StoreClient(
    base_url="https://dashboard.snapcraft.io",
    storage_base_url="https://upload.apps.staging.ubuntu.com",
    endpoints=endpoints.SNAP_STORE,
    user_agent="Craft Store Tutorial Agent",
    application_name="cart-store-tutorial"
)

whoami = store_client.whoami()

print(f"email: {whoami['account']['email']}")
print(f"id: {whoami['account']['id']}")
```

1.3.3 Run

Run the saved python module to retrieved the account information for the login:

```
$ python snap_store_whoami.py
```

1.4 Upload a snap to storage

At the end of this tutorial you will be able to upload a snap to file storage and see simple progress and total length updated on the screen as the upload takes place.

1.4.1 Prerequisites

- Python 3.8
- a clean virtual environment setup
- a text editor

1.4.2 Setup

Create a clean virtual environment:

```
$ pip3 -m venv ~/craft-store-upload
$ . ~/craft-store-upload/bin/activate
```

Install Craft Store by running:

```
$ pip install craft-store
```

Obtain a snap to upload by downloading one from the Snap Store and give it a predictable name:

```
$ snap download hello
$ mv hello_*.snap /tmp/hello.snap
```

1.4.3 Code for uploading

Open a text editor to add logic to instantiate a StoreClient for the Staging Snap Store:

```
#!/usr/bin/env python
from pathlib import Path

from craft_store import StoreClient, endpoints

store_client = StoreClient(
    base_url="https://dashboard.staging.snapcraft.io",
    storage_base_url="https://upload.apps.staging.ubuntu.com",
    endpoints=endpoints.SNAP_STORE,
    user_agent="Craft Store Tutorial Agent",
    application_name="craft-store-tutorial"
)

upload_id = store_client.upload_file(filepath=Path("/tmp/hello.snap"))

print(f"upload-id: {upload_id}")
```

Save the file as `snap_store_upload.py`:

1.4.4 Run

Run the saved python module to upload the *hello* snap and obtain an upload-id at the end:

```
$ python snap_store_upload.py
```

1.4.5 Adding progress

Now add a mechanism to view progress for the upload, open the recently saved `snap_store_upload.py` file and modify it so that it looks like the following:

```
#!/usr/bin/env python
from pathlib import Path

from craft_store import StoreClient, endpoints
from requests_toolbelt import MultipartEncoder, MultipartEncoderMonitor


def monitor_callback(encoder: MultipartEncoder) -> None:
    def progress_callback(monitor: MultipartEncoderMonitor) -> None:
        print(f"Uploaded: {monitor.bytes_read} of {monitor.len}")
```

(continues on next page)

(continued from previous page)

```
    return progress_callback


store_client = StoreClient(
    base_url="https://dashboard.staging.snapcraft.io",
    storage_base_url="https://upload.apps.staging.ubuntu.com",
    endpoints=endpoints.SNAP_STORE,
    user_agent="Craft Store Tutorial Agent",
    application_name="craft-store-tutorial"
)

upload_id = store_client.upload_file(
     filepath=Path("/tmp/hello.snap"),
     monitor_callback=progress_callback
)

print(f"upload-id: {upload_id}")
```

Save the file.

1.4.6 Run

Run the saved python module again to upload the *hello* snap and obtain an upload-id at the end, but observing progress as the upload takes place:

```
$ python snap_store_upload.py
```

CHAPTER

TWO

HOW TO

2.1 Using credentials provided by an environment variable

2.1.1 Retrieving the credentials

Use the following snippet to obtain general credentials for your account:

```
#!/usr/bin/env python
from craft_store import StoreClient, endpoints

store_client = StoreClient(
    base_url="https://dashboard.snapcraft.io",
    endpoints=endpoints.SNAP_STORE,
    user_agent="Craft Store Tutorial Agent",
    application_name="cart-store-tutorial"
)

store_client.login(
    permissions=["package_access"],
    description="tutorial-client-login",
    ttl=1000
)

print(f"Exported credentials: {credentials}")
```

Note: The `craft_store.store_client.StoreClient.login()` method has some extra parameters such as `packages` and `channels` to restrict the credentials *reach* even further. Also take consideration into further locking down `permissions` (*craft_store.attenuations*).

2.1.2 Using retrieved credentials

If *craft_store.store_client.StoreClient* is initialized with *environment_auth* and the value is set then a in-memory keyring is used instead of the system keyring.

To make use of such thing, export CREDENTIALS=<credentials> where <credentials> is the recently retrieved credential. To make use of it and get information from your account:

```
#!/usr/bin/env python
from craft_store import StoreClient, endpoints

store_client = StoreClient(
    base_url="https://dashboard.snapcraft.io",
    endpoints=endpoints.SNAP_STORE,
    user_agent="Craft Store Tutorial Agent",
    application_name="cart-store-tutorial",
    environment_auth="CREDENTIALS",
)

whoami = store_client.whoami()

print(f"email: {whoami['account']['email']}")
print(f"id: {whoami['account']['id']}")
```

2.2 Using craft-cli for upload progress

Progress can be provided by use of craft-cli. This example will upload *./test.snap* with something that looks like the following:

```
#!/usr/bin/env python
from pathlib import Path

from craft_cli import emit, EmitterMode
from craft_store import StoreClient, endpoints
from requests_toolbelt import MultipartEncoder, MultipartEncoderMonitor


emit.init(EmitterMode.NORMAL, "craft.store-howto", "Starting howto app.")

store_client = StoreClient(
    base_url="https://dashboard.staging.snapcraft.io",
    storage_base_url="https://upload.apps.staging.ubuntu.com",
    endpoints=endpoints.SNAP_STORE,
    user_agent="Craft Store Howto Agent",
    application_name="craft-store-upload",
)


def create_callback(encoder: MultipartEncoder):
    with emit.progress_bar("Uploading...", encoder.len, delta=False) as progress:

        def progress_callback(monitor: MultipartEncoderMonitor):
```

(continues on next page)

(continued from previous page)

```
            progress.advance(monitor.bytes_read)

    return progress_callback


upload_id = store_client.upload_file(
    monitor_callback=create_callback,
    filepath=Path("test.snap"),
)

emit.message(f"upload-id: {upload_id}", intermediate=True)
emit.ended_ok()
```

CHAPTER

THREE

CRAFT_STORE PACKAGE

3.1 Subpackages

3.1.1 craft_store.models package

Submodules

craft_store.models.charm_list_releases_model module

List Releases Models for Charmhub responses.

class craft_store.models.charm_list_releases_model.**ChannelMapModel**(***data*)

Bases: *craft_store.models._base_model.MarshableModel*

Model for the channel-map results from the list_releases endpoint.

Parameters **data** (Any) –

base: *craft_store.models.charm_list_releases_model.CharmBaseModel*

channel: str

expiration_date: Optional[datetime.datetime]

progressive: craft_store.models._common_list_releases_model.ProgressiveModel

resources: List[*craft_store.models.charm_list_releases_model.ResourceModel*]

revision: int

when: datetime.datetime

class craft_store.models.charm_list_releases_model.**CharmBaseModel**(***data*)

Bases: *craft_store.models._base_model.MarshableModel*

Base entries for the channel-map entry from the list_releases endpoint.

Parameters **data** (Any) –

architecture: str

channel: str

name: str

class craft_store.models.charm_list_releases_model.**ListReleasesModel**(***data*)

Bases: *craft_store.models._base_model.MarshableModel*

Model for the list_releases endpoint.

Parameters **data** (Any) –

channel_map: List[*craft_store.models.charm_list_releases_model.ChannelMapModel*]

package: craft_store.models._common_list_releases_model.PackageModel

revisions: List[*craft_store.models.charm_list_releases_model.RevisionModel*]

class craft_store.models.charm_list_releases_model.**ResourceModel**(***data*)
Bases: *craft_store.models._base_model.MarshableModel*

Resource entries for the channel-map entry from the list_releases endpoint.

Parameters **data** (Any) –

name: str

revision: Optional[int]

type: str

class craft_store.models.charm_list_releases_model.**RevisionModel**(***data*)
Bases: *craft_store.models._base_model.MarshableModel*

Model for a revision entry from list_releases.

Parameters **data** (Any) –

bases: List[*craft_store.models.charm_list_releases_model.CharmBaseModel*]

created_at: datetime.datetime

errors: Any

revision: int

sha3_384: str

size: int

status: str

version: str

craft_store.models.release_request_model module

Release request models for the Store.

class craft_store.models.release_request_model.**ReleaseRequestModel**(***data*)
Bases: *craft_store.models._base_model.MarshableModel*

Model to request a release to the store.

Parameters

- **channel** – name of the channel to release to.
- **resources** – resources to release with this revision.
- **revision** – revision to release.
- **data** (Any) –

channel: str

resources: Optional[List[*craft_store.models.release_request_model.ResourceModel*]]

revision: Optional[int]

class craft_store.models.release_request_model.**ResourceModel**(**data)
Bases: craft_store.models._base_model.MarshableModel

Resource model for a ReleaseRequestModel.

Parameters

- **name** – name of the resource to release.
- **revision** – revision of the resource to release.
- **data** (Any) –

name: str

revision: Optional[int]

craft_store.models.revisions_model module

Revisions response models for the Store.

class craft_store.models.revisions_model.**RevisionsRequestModel**(**data)
Bases: craft_store.models._base_model.MarshableModel

Resource model for a ReleaseRequestModel.

Parameters

- **upload_id** – the upload-id returned from the storage endpoint.
- **data** (Any) –

upload_id: str

class craft_store.models.revisions_model.**RevisionsResponseModel**(**data)
Bases: craft_store.models._base_model.MarshableModel

Model for a revisions response.

Parameters

- **status-url** – a URL to monitor the state of the posted revision.
- **data** (Any) –

status_url: str

craft_store.models.snap_list_releases_model module

List Releases Models for Snap Store responses.

class craft_store.models.snap_list_releases_model.**ChannelMapModel**(**data)
Bases: craft_store.models._base_model.MarshableModel

Model for the channel-map results from the list_releases endpoint.

Parameters **data** (Any) –

architecture: str

channel: str

expiration_date: Optional[datetime.datetime]

progressive: craft_store.models._common_list_releases_model.ProgressiveModel

revision: int

when: datetime.datetime

class craft_store.models.snap_list_releases_model.**Confinement**(*value*)
Bases: str, enum.Enum

Snap confinement.

CLASSIC = 'classic'

DEVMODE = 'devmode'

STRICT = 'strict'

class craft_store.models.snap_list_releases_model.**Grade**(*value*)
Bases: str, enum.Enum

Snap grade.

DEVEL = 'devel'

STABLE = 'stable'

class craft_store.models.snap_list_releases_model.**ListReleasesModel**(***data*)
Bases: *craft_store.models._base_model.MarshableModel*

Model for the list_releases endpoint.

Parameters **data** (Any) –

channel_map: List[*craft_store.models.snap_list_releases_model.ChannelMapModel*]

package: craft_store.models._common_list_releases_model.PackageModel

revisions: List[*craft_store.models.snap_list_releases_model.RevisionModel*]

class craft_store.models.snap_list_releases_model.**RevisionModel**(***data*)
Bases: *craft_store.models._base_model.MarshableModel*

Model for a revision entry from list_releases.

Parameters **data** (Any) –

architectures: List[str]

base: Optional[str]

build_url: Optional[str]

confinement: *craft_store.models.snap_list_releases_model.Confinement*

created_at: datetime.datetime

created_by: str

grade: *craft_store.models.snap_list_releases_model.Grade*

revision: int

sha3_384: str

size: int

status: str

type: *craft_store.models.snap_list_releases_model.Type*

version: str

class craft_store.models.snap_list_releases_model.**Type**(*value*)
Bases: str, enum.Enum

Type of snap.

APP = 'app'

BASE = 'base'

GADGET = 'gadget'

KERNEL = 'kernel'

SNAPD = 'snapd'

Module contents

Models package for store responses.

craft_store.models.**CharmListReleasesModel**
alias of *craft_store.models.charm_list_releases_model.ListReleasesModel*

class craft_store.models.**MarshableModel**(***data*)
Bases: pydantic.main.BaseModel

A BaseModel that can be marshaled and unmarshaled.

Parameters **data** (Any) –

class Config
Bases: object

Pydantic model configuration.

alias_generator()

allow_mutation = False

validate_assignment = True

marshal()
Create a dictionary containing the part specification data.

Return type Dict[str, Any]

Returns The newly created dictionary.

classmethod unmarshal(*data*)
Create and populate a new MarshableModel from a dict.

The unmarshal method validates entries in the input dictionary, populating the corresponding fields in the data object.

Parameters **data** (Dict[str, Any]) – The dictionary data to unmarshal.

Return type *MarshableModel*

Returns The newly created object.

Raises **TypeError** – If data is not a dictionary.

class craft_store.models.**ReleaseRequestModel**(***data*)
Bases: *craft_store.models._base_model.MarshableModel*

Model to request a release to the store.

Parameters

- **channel** – name of the channel to release to.
- **resources** – resources to release with this revision.
- **revision** – revision to release.
- **data** (Any) –

channel: str

resources: Optional[List[*craft_store.models.release_request_model.ResourceModel*]]

revision: Optional[int]

class craft_store.models.**RevisionsRequestModel**(***data*)
Bases: *craft_store.models._base_model.MarshableModel*

Resource model for a ReleaseRequestModel.

Parameters

- **upload_id** – the upload-id returned from the storage endpoint.
- **data** (Any) –

upload_id: str

class craft_store.models.**RevisionsResponseModel**(***data*)
Bases: *craft_store.models._base_model.MarshableModel*

Model for a revisions response.

Parameters

- **status-url** – a URL to monitor the state of the posted revision.
- **data** (Any) –

status_url: str

craft_store.models.**SnapListReleasesModel**
alias of *craft_store.models.snap_list_releases_model.ListReleasesModel*

3.2 Submodules

3.2.1 craft_store.attenuations module

Attenuations for Snap Store and Charmhub discharged Macaroons.

craft_store.attenuations.**ACCOUNT_REGISTER_PACKAGE = 'account-register-package'**
Register or request a new package name under a given account.

craft_store.attenuations.**ACCOUNT_VIEW_PACKAGES = 'account-view-packages'**
List packages owned by the account and packages for which this account has collaborator rights.

craft_store.attenuations.**PACKAGE_MANAGE = 'package-manage'**
Meta permission for easing creation of a complete RW token, it grants all the package-manage-* permissions.

craft_store.attenuations.**PACKAGE_MANAGE_ACL = 'package-manage-acl'**
Add, invite or remove collaborators.

craft_store.attenuations.**PACKAGE_MANAGE_LIBRARY = 'package-manage-library'**
Register or request a new library name under a given package.

craft_store.attenuations.**PACKAGE_MANAGE_METADATA = 'package-manage-metadata'**
Edit metadata, add or remove media, etc.

craft_store.attenuations.**PACKAGE_MANAGE_RELEASES = 'package-manage-releases'**
Release revisions, close channels and update version pattern for a track.

craft_store.attenuations.**PACKAGE_MANAGE_REVISIONS = 'package-manage-revisions'**
Upload new blobs, check for upload status, reject a revision blocked on manual review or request manual review.

craft_store.attenuations.**PACKAGE_VIEW = 'package-view'**
Meta permission for easing creation of a complete read only token grants all the package-view-* permissions.

craft_store.attenuations.**PACKAGE_VIEW_ACL = 'package-view-acl'**
List the collaborators for a package and privacy settings.

craft_store.attenuations.**PACKAGE_VIEW_METADATA = 'package-view-metadata'**
View the metadata for a package, including media.

craft_store.attenuations.**PACKAGE_VIEW_METRICS = 'package-view-metrics'**
View the metrics of a package.

craft_store.attenuations.**PACKAGE_VIEW_RELEASES = 'package-view-releases'**
List the current releases (channel map) for a package and the release history of a package.

craft_store.attenuations.**PACKAGE_VIEW_REVISIONS = 'package-view-revisions'**
List the existing revisions for a package, along with status information.

3.2.2 craft_store.auth module

Craft Store Authentication Store.

class craft_store.auth.**Auth**(*application_name*, *host*, *environment_auth=None*, *ephemeral=False*)
Bases: object

Auth wraps around the keyring to store credentials.

The application_name and host are used as key/values in the keyring to set, get and delete credentials.

If environment_auth is set on initialization of this class, then a *MemoryKeyring* is setup in lieu of the system one.

Credentials are base64 encoded into the keyring and decoded on retrieval.

Variables

- **application_name** – name of the application using this library.
- **host** – specific host for the store used.

Parameters

- **application_name** (str) –
- **host** (str) –
- **environment_auth** (Optional[str]) –
- **ephemeral** (bool) –

static decode_credentials(*encoded_credentials*)
Decode base64 encoded credentials.

Raises *errors.CredentialsNotParseable* – when the credentials are incorrectly encoded.

Parameters **encoded_credentials** (str) –

Return type str

del_credentials()
Delete credentials from the keyring.

Return type None

static encode_credentials(*credentials*)
Encode credentials to base64.

Parameters **credentials** (str) –

Return type str

ensure_no_credentials()
Check that no credentials exist.

Raises **errors.CredentialsAvailable** – if credentials have already been set.

Return type None

get_credentials()
Retrieve credentials from the keyring.

Return type str

set_credentials(*credentials*, *force=False*)
Store credentials in the keyring.

Parameters

- **credentials** (str) – token to store.
- **force** (bool) – overwrite existing credentials.

Return type None

class craft_store.auth.MemoryKeyring
Bases: keyring.backend.KeyringBackend

A keyring that stores credentials in a dictionary.

delete_password(*service*, *username*)
Delete the service password for username from memory.

Parameters

- **service** (str) –
- **username** (str) –

Return type None

get_password(*service*, *username*)
Get the service password for username from memory.

Parameters

- **service** (str) –
- **username** (str) –

Return type Optional[str]

priority = -1

set_password(*service*, *username*, *password*)
Set the service password for username in memory.

Parameters

- **service** (str) –
- **username** (str) –
- **password** (str) –

Return type None

3.2.3 craft_store.base_client module

Craft Store BaseClient.

class craft_store.base_client.**BaseClient**(*, *base_url*, *storage_base_url*, *endpoints*, *application_name*, *user_agent*, *environment_auth=None*, *ephemeral=False*)

Bases: object

Encapsulates API calls for the Snap Store or Charmhub.

Parameters

- **base_url** (str) – the base url of the API endpoint.
- **storage_base_url** (str) – the base url for storage.
- **endpoints** (*Endpoints*) – *endpoints.CHARMHUB* or *endpoints.SNAP_STORE*.
- **application_name** (str) – the name application using this class, used for the keyring.
- **user_agent** (str) – User-Agent header to use for HTTP(s) requests.
- **environment_auth** (Optional[str]) – environment variable to use for credentials.
- **ephemeral** (bool) – keep everything in memory.

Raises *errors.NoKeyringError* – if there is no usable keyring.

get_list_releases(*, *name*)

Query the list_releases endpoint and return the result.

Parameters **name** (str) –

Return type *MarshableModel*

login(*, *permissions*, *description*, *ttl*, *packages=None*, *channels=None*, ***kwargs*)

Obtain credentials to perform authenticated requests.

Credentials are stored on the system's keyring, handled by *craft_store.auth.Auth*.

The list of permissions to select from can be referred to on *craft_store.attenuations*.

The login process requires 3 steps:

- request an initial macaroon on *endpoints.Endpoints.tokens*.
- discharge that macaroon using Candid
- send the discharge macaroon to *endpoints.Endpoints.tokens_exchange* to obtain final authorization of the macaroon

This last macaroon is stored into the system's keyring to perform authenticated requests.

Parameters

- **permissions** (Sequence[str]) – Set of permissions to grant the login.
- **description** (str) – Client description to refer to from the Store.

- **ttl** (int) – time to live for the credential, in other words, how long until it expires, expressed in seconds.
- **packages** (Optional[Sequence[*Package*]]) – Sequence of packages to limit the credentials to.
- **channels** (Optional[Sequence[str]]) – Sequence of channel names to limit the credentials to.

Raises *errors.CredentialsAlreadyAvailable* – if credentials already exist.

Return type str

logout()
Clear credentials.

Raises *errors.CredentialsUnavailable* – if credentials cannot be found.

Return type None

notify_revision(*, *name*, *revision_request*)
Post to the revisions endpoint to notify the store about an upload.

This request usually takes place after a successful *upload*.

Parameters

- **name** (str) –
- **revision_request** (*RevisionsRequestModel*) –

Return type *RevisionsResponseModel*

release(*, *name*, *release_request*)
Request a release of name.

Parameters

- **name** (str) – name to release.
- **release_request** (Sequence[*ReleaseRequestModel*]) – sequence of items to release.

Return type None

request(*method*, *url*, *params=None*, *headers=None*, ***kwargs*)
Perform an authenticated request if auth_headers are True.

Parameters

- **method** (str) – HTTP method used for the request.
- **url** (str) – URL to request with method.
- **params** (Optional[Dict[str, str]]) – Query parameters to be sent along with the request.
- **headers** (Optional[Dict[str, str]]) – Headers to be sent along with the request.

Raises

- *errors.StoreServerError* – for error responses.
- *errors.NetworkError* – for lower level network issues.
- *errors.CredentialsUnavailable* – if credentials cannot be found.

Return type Response

Returns Response from the request.

upload_file(*, *filepath*, *monitor_callback=None*)
Upload filepath to storage.

The monitor_callback is a method receiving one argument of type `MultipartEncoder`, the total length of the upload can be accessed from this encoder from the `len` attribute to setup a progress bar instance.

The callback is to return a function that receives a `MultipartEncoderMonitor` from which the `.bytes_read` attribute can be read to update progress.

The simplest implementation can look like:

```
def monitor_callback(encoder: requests_toolbelt.MultipartEncoder):

    # instantiate progress class with total bytes encoder.len

    def progress_printer(monitor: requests_toolbelt.MultipartEncoderMonitor):
       # Print progress using monitor.bytes_read

    return progress_printer
```

Parameters

- **monitor_callback** (Optional[Callable]) – a callback to monitor progress.
- **filepath** (Path) –

Return type str

whoami()
Return whoami json data queyring *endpoints.Endpoints.whoami*.

Return type Dict[str, Any]

3.2.4 craft_store.creds module

Functions to serialize/deserialize credentials for Candid and Ubuntu One SSO.

class craft_store.creds.**CandidModel**(***data*)
Bases: pydantic.main.BaseModel

Model for Candid credentials.

Parameters **data** (Any) –

marshal()
Create a dictionary containing the Candid credentials.

Return type Dict[str, Any]

token_type: Literal['macaroon']

classmethod unmarshal(*data*)
Create Candid model from dictionary data.

Parameters **data** (Dict[str, Any]) –

Return type *CandidModel*

value: str

class craft_store.creds.**UbuntuOneMacaroons**(***data*)
Bases: pydantic.main.BaseModel

Model representation of the set of macaroons used in Ubuntu SSO.

Parameters **data** (Any) –

discharge: str

root: str

with_discharge(*discharge*)
Create a copy of this UbuntuOneMacaroons with a different discharge macaroon.

Parameters **discharge** (str) –

Return type *UbuntuOneMacaroons*

class craft_store.creds.**UbuntuOneModel**(***data*)
Bases: pydantic.main.BaseModel

Model for Ubuntu One credentials.

Parameters **data** (Any) –

marshal()
Create a dictionary containing the Ubuntu One credentials.

Return type Dict[str, Any]

token_type: Literal['u1-macaroon']

classmethod unmarshal(*data*)
Create Candid model from dictionary data.

Parameters **data** (Dict[str, Any]) –

Return type *UbuntuOneModel*

value: *craft_store.creds.UbuntuOneMacaroons*

craft_store.creds.**marshal_candid_credentials**(*candid_creds*)
Serialize Candid credentials for storage.

This function creates a string that contains the desired *candid_creds* but also stores their "type", for unmarshalling later with *unmarshal_candid_credentials()*.

Parameters **candid_creds** (str) – The actual Candid credentials.

Return type str

Returns A payload string ready to be passed to Auth.set_credentials()

craft_store.creds.**marshal_u1_credentials**(*u1_creds*)
Serialize Ubuntu One credentials for storage.

This function creates a string that contains the desired *u1_creds* but also stores their "type", for unmarshalling later with *unmarshal_u1_credentials()*.

Parameters **u1_creds** (*UbuntuOneMacaroons*) – The actual Ubuntu One macaroons credentials.

Return type str

Returns A payload string ready to be passed to Auth.set_credentials()

craft_store.creds.**unmarshal_candid_credentials**(*marshalled_creds*)
Deserialize previously stored Candid credentials.

This function is meant to be called with the returned value from Auth.get_credentials(). As such, it supports parsing credentials from before we stored the *token_type*. Overall, this means supporting the scenarios where *marshalled_creds* is:

(1) a regular JSON string generated by *marshal_candid_credentials()*;

(2) a regular JSON string that was *not* generated by *marshal_candid_credentials()* but contains the Candid macaroon as a serialized dict;

(3) a string that is not JSON but contains the serialized (non-dict) Candid macaroon.

In addition, the function will raise a CredentialsNotParseable error when *marshalled_creds* is:

(4) A regular JSON string containing a dict that *has* a type, but is *not* Candid.

Parameters **marshalled_creds** (str) – The credentials retrieved from auth storage.

Return type str

Returns The actual Candid credentials.

craft_store.creds.**unmarshal_u1_credentials**(*marshalled_creds*)

Deserialize previously stored Ubuntu One credentials.

This function is meant to be called with the returned value from Auth.get_credentials(). As such, it supports parsing credentials from before we stored the *token_type*. Overall, this means supporting the scenarios where *marshalled_creds* is:

(1) a regular JSON string generated by *marshal_u1_credentials()*;

(2) a regular JSON string that was *not* generated by *marshal_u1_credentials()* but contains the *UbuntuOneMacaroons* values as a serialized dict;

In addition, the function will raise a CredentialsNotParseable error when *marshalled_creds* is:

(3) a string that is not JSON.

(4) A regular JSON string containing a dict that *has* a type, but is *not* Ubuntu One.

Parameters **marshalled_creds** (str) – The credentials retrieved from auth storage.

Return type *UbuntuOneMacaroons*

Returns The actual Ubuntu One credentials.

3.2.5 craft_store.endpoints module

Endpoint definitions for different services.

craft_store.endpoints.**CHARMHUB: Final = Endpoints(namespace='charm', whoami='/v1/tokens/whoami', tokens='/v1/tokens', tokens_exchange='/v1/tokens/exchange', list_releases_model=<class 'craft_store.models.charm_list_releases_model.ListReleasesModel'>, valid_package_types=['charm', 'bundle'], upload='/unscanned-upload/', tokens_refresh=None)**

Charmhub set of supported endpoints.

class craft_store.endpoints.**Endpoints**(*namespace*, *whoami*, *tokens*, *tokens_exchange*, *list_releases_model*, *valid_package_types*, *upload='/unscanned-upload/'*, *tokens_refresh=None*)

Bases: object

Endpoints used to make requests to a store.

Parameters

- **namespace** (str) – the namespace to use for endpoints.
- **whoami** (str) – path to the whoami API.
- **upload** (str) – path used for uploading.
- **tokens** (str) – path to the tokens API.
- **tokens_exchange** (str) – path to the tokens_exchange API.
- **tokens_refresh** (Optional[str]) – path to the tokens_refresh API.
- **list_releases_model** (Type[*MarshableModel*]) – list_releases response model.
- **valid_package_types** (Sequence[str]) –

get_releases_endpoint(*name*)

Return the slug to the releases endpoint.

Parameters **name** (str) –

Return type str

get_revisions_endpoint(*name*)

Return the slug to the revisions endpoint.

Parameters **name** (str) –

Return type str

get_token_request(*, *permissions*, *description*, *ttl*, *channels=None*, *packages=None*)

Return a properly formatted request for a token request.

Permissions can be selected from *craft_store.attenuations*

Parameters

- **permissions** (Sequence[str]) – a list of permissions to use.
- **description** (str) – description that identifies the client.
- **ttl** (int) – time to live for the requested token.
- **packages** (Optional[Sequence[*Package*]]) – a sequence of *Package* to limit the requested token to.
- **channels** (Optional[Sequence[str]]) – a sequence of channels to limit the requested token to.

Return type Dict[str, Any]

static get_upload_id(*result*)

Return the upload ID for a given result.

Parameters **result** (Dict[str, Any]) – the result from an upload request.

Return type str

list_releases_model: Type[*craft_store.models._base_model.MarshableModel*]

namespace: str

tokens: str

tokens_exchange: str

tokens_refresh: Optional[str] = None

upload: str = '/unscanned-upload/'

valid_package_types: Sequence[str]

whoami: str

class craft_store.endpoints.**Package**(*package_name*, *package_type*)
Bases: object

Representation of a package name and type.

Parameters

- **package_name** (str) –
- **package_type** (str) –

package_name: str

package_type: str

craft_store.endpoints.**SNAP_STORE: Final = _SnapStoreEndpoints(namespace='snap', whoami='/api/v2/tokens/whoami', tokens='/api/v2/tokens', tokens_exchange='/api/v2/tokens/exchange', list_releases_model=<class 'craft_store.models.snap_list_releases_model.ListReleasesModel'>, valid_package_types=['snap'], upload='/unscanned-upload/', tokens_refresh=None)**
Snap Store set of supported endpoints.

craft_store.endpoints.**U1_SNAP_STORE: Final = _SnapStoreEndpoints(namespace='snap', whoami='/api/v2/tokens/whoami', tokens='/dev/api/acl/', tokens_exchange='/api/v2/tokens/discharge', list_releases_model=<class 'craft_store.models.snap_list_releases_model.ListReleasesModel'>, valid_package_types=['snap'], upload='/unscanned-upload/', tokens_refresh='/api/v2/tokens/refresh')**
Ubuntu One compatible Snap Store set of supported endpoints.

3.2.6 craft_store.errors module

Craft Store errors.

exception craft_store.errors.**CandidTokenKindError**(*url*)
Bases: *craft_store.errors.CraftStoreError*

Error raised when the token kind is missing from the discharged macaroon.

Parameters **url** (str) –

exception craft_store.errors.**CandidTokenTimeoutError**(*url*)
Bases: *craft_store.errors.CraftStoreError*

Error raised when timeout is reached trying to discharge a macaroon.

Parameters **url** (str) –

exception craft_store.errors.**CandidTokenValueError**(*url*)
Bases: *craft_store.errors.CraftStoreError*

Error raised when the token value is missing from the discharged macaroon.

Parameters **url** (str) –

exception craft_store.errors.**CraftStoreError**(*message*, *resolution=None*)
Bases: Exception

Base class error for craft-store.

Parameters

- **message** (str) –
- **resolution** (Optional[str]) –

exception craft_store.errors.**CredentialsAlreadyAvailable**(*application*, *host*)
Bases: *craft_store.errors.CraftStoreError*

Error raised when credentials are already found in the keyring.

Parameters

- **application** (str) –
- **host** (str) –

exception craft_store.errors.**CredentialsNotParseable**(*msg='Expected base64 encoded credentials'*)
Bases: *craft_store.errors.CraftStoreError*

Error raised when credentials are not parseable.

Parameters **msg** (str) –

exception craft_store.errors.**CredentialsUnavailable**(*application*, *host*)
Bases: *craft_store.errors.CraftStoreError*

Error raised when credentials are not found in the keyring.

Parameters

- **application** (str) –
- **host** (str) –

exception craft_store.errors.**NetworkError**(*exception*)
Bases: *craft_store.errors.CraftStoreError*

Error to raise on network or infrastructure issues.

The original exception is used to potentially craft a user friendly error message to be used for `brief`.

Parameters **exception** (Exception) – original exception raised.

Variables **exception** – original exception raised.

exception craft_store.errors.**NoKeyringError**
Bases: *craft_store.errors.CraftStoreError*

Error raised when no keyring can be used.

class craft_store.errors.**StoreErrorList**(*error_list*)
Bases: object

Error List returned from the Store.

Parameters **error_list** (List[Dict[str, str]]) –

exception craft_store.errors.**StoreServerError**(*response*)
Bases: *craft_store.errors.CraftStoreError*

Error to raise on infrastructure issues from error codes above `500`.

Parameters **response** (Response) – the response from a `requests.Request`.

Variables

- **response** – the response from a requests.Request.
- **error_list** – list of errors returned by the Store *StoreErrorList*.

3.2.7 craft_store.http_client module

Craft Store HTTPClient.

class craft_store.http_client.**HTTPClient**(*, *user_agent*)

Bases: object

Generic HTTP Client to communicate with Canonical's Developer Gateway.

This client has a requests like interface, it creates a requests.Session on initialization to handle retries over HTTP and HTTPS requests.

The default number of retries is set in *REQUEST_TOTAL_RETRIES* and can be overridden with the CRAFT_STORE_RETRIES environment variable.

The backoff factor has a default set in *REQUEST_BACKOFF* and can be overridden with the CRAFT_STORE_BACKOFF environment variable.

Retries are done for the following return codes: 500, 502, 503 and 504.

Variables **user_agent** – User-Agent header to identify the client.

Parameters **user_agent** (str) –

get(**args*, ***kwargs*)

Perform an HTTP GET request.

Return type Response

post(**args*, ***kwargs*)

Perform an HTTP POST request.

Return type Response

put(**args*, ***kwargs*)

Perform an HTTP PUT request.

Return type Response

request(*method*, *url*, *params=None*, *headers=None*, ***kwargs*)

Send a request to url.

user_agent is set as part of the headers for the request. All requests are logged through a debug logs, headers matching Authorization and Macaroons have their value replaced.

Parameters

- **method** (str) – HTTP method used for the request.
- **url** (str) – URL to request with method.
- **params** (Optional[Dict[str, str]]) – Query parameters to be sent along with the request.
- **headers** (Optional[Dict[str, str]]) – Headers to be sent along with the request.

Raises

- *errors.StoreServerError* – for error responses.

- *errors.NetworkError* – for lower level network issues.

Return type Response

Returns Response from the request.

craft_store.http_client.**REQUEST_BACKOFF = 1**
Backoff before retrying a request.

craft_store.http_client.**REQUEST_TOTAL_RETRIES = 8**
Amount of retries for a request.

3.2.8 craft_store.store_client module

Craft Store StoreClient.

class craft_store.store_client.**StoreClient**(*, *base_url*, *storage_base_url*, *endpoints*, *application_name*, *user_agent*, *environment_auth=None*, *ephemeral=False*)
Bases: *craft_store.base_client.BaseClient*

Encapsulates API calls for the Snap Store or Charmhub.

Parameters

- **base_url** (str) –
- **storage_base_url** (str) –
- **endpoints** (*Endpoints*) –
- **application_name** (str) –
- **user_agent** (str) –
- **environment_auth** (Optional[str]) –
- **ephemeral** (bool) –

TOKEN_TYPE: str = 'macaroon'

class craft_store.store_client.**WebBrowserWaitingInteractor**(*user_agent*)
Bases: macaroonbakery.httpbakery._browser.WebBrowserInteractor

WebBrowserInteractor implementation using HTTPClient.

Waiting for a token is implemented using HTTPClient which mounts a session with backoff retries.

Better exception classes and messages are provided to handle errors.

Parameters **user_agent** (str) –

3.2.9 craft_store.ubuntu_one_store_client module

Craft Store StoreClient.

class craft_store.ubuntu_one_store_client.**UbuntuOneStoreClient**(*, *base_url*, *storage_base_url*, *auth_url*, *endpoints*, *application_name*, *user_agent*, *environment_auth=None*, *ephemeral=False*)
Bases: *craft_store.base_client.BaseClient*

Encapsulates API calls for the Snap Store or Charmhub with Ubuntu One.

Parameters

- **base_url** (str) –
- **storage_base_url** (str) –
- **auth_url** (str) –
- **endpoints** (*Endpoints*) –
- **application_name** (str) –
- **user_agent** (str) –
- **environment_auth** (Optional[str]) –
- **ephemeral** (bool) –

TOKEN_TYPE: str = 'u1-macaroon'

request(*method*, *url*, *params=None*, *headers=None*, ***kwargs*)

Perform an authenticated request if auth_headers are True.

Parameters

- **method** (str) – HTTP method used for the request.
- **url** (str) – URL to request with method.
- **params** (Optional[Dict[str, str]]) – Query parameters to be sent along with the request.
- **headers** (Optional[Dict[str, str]]) – Headers to be sent along with the request.

Raises

- *errors.StoreServerError* – for error responses.
- *errors.NetworkError* – for lower level network issues.
- *errors.CredentialsUnavailable* – if credentials cannot be found.

Return type Response

Returns Response from the request.

3.3 Module contents

Interact with Canonical services such as Charmhub and the Snap Store.

class craft_store.**Auth**(*application_name*, *host*, *environment_auth=None*, *ephemeral=False*)

Bases: object

Auth wraps around the keyring to store credentials.

The application_name and host are used as key/values in the keyring to set, get and delete credentials.

If environment_auth is set on initialization of this class, then a MemoryKeyring is setup in lieu of the system one.

Credentials are base64 encoded into the keyring and decoded on retrieval.

Variables

- **application_name** – name of the application using this library.
- **host** – specific host for the store used.

Parameters

- **application_name** (str) –
- **host** (str) –
- **environment_auth** (Optional[str]) –
- **ephemeral** (bool) –

static decode_credentials(*encoded_credentials*)

Decode base64 encoded credentials.

Raises *errors.CredentialsNotParseable* – when the credentials are incorrectly encoded.

Parameters **encoded_credentials** (str) –

Return type str

del_credentials()

Delete credentials from the keyring.

Return type None

static encode_credentials(*credentials*)

Encode credentials to base64.

Parameters **credentials** (str) –

Return type str

ensure_no_credentials()

Check that no credentials exist.

Raises **errors.CredentialsAvailable** – if credentials have already been set.

Return type None

get_credentials()

Retrieve credentials from the keyring.

Return type str

set_credentials(*credentials*, *force=False*)

Store credentials in the keyring.

Parameters

- **credentials** (str) – token to store.
- **force** (bool) – overwrite existing credentials.

Return type None

class craft_store.**BaseClient**(*, *base_url*, *storage_base_url*, *endpoints*, *application_name*, *user_agent*, *environment_auth=None*, *ephemeral=False*)

Bases: object

Encapsulates API calls for the Snap Store or Charmhub.

Parameters

- **base_url** (str) – the base url of the API endpoint.
- **storage_base_url** (str) – the base url for storage.
- **endpoints** (*Endpoints*) – *endpoints.CHARMHUB* or *endpoints.SNAP_STORE*.
- **application_name** (str) – the name application using this class, used for the keyring.

- **user_agent** (str) – User-Agent header to use for HTTP(s) requests.
- **environment_auth** (Optional[str]) – environment variable to use for credentials.
- **ephemeral** (bool) – keep everything in memory.

Raises *errors.NoKeyringError* – if there is no usable keyring.

get_list_releases(*, *name*)

Query the list_releases endpoint and return the result.

Parameters **name** (str) –

Return type *MarshableModel*

login(*, *permissions*, *description*, *ttl*, *packages=None*, *channels=None*, ***kwargs*)

Obtain credentials to perform authenticated requests.

Credentials are stored on the system's keyring, handled by *craft_store.auth.Auth*.

The list of permissions to select from can be referred to on *craft_store.attenuations*.

The login process requires 3 steps:

- request an initial macaroon on *endpoints.Endpoints.tokens*.
- discharge that macaroon using Candid
- send the discharge macaroon to *endpoints.Endpoints.tokens_exchange* to obtain final authorization of the macaroon

This last macaroon is stored into the system's keyring to perform authenticated requests.

Parameters

- **permissions** (Sequence[str]) – Set of permissions to grant the login.
- **description** (str) – Client description to refer to from the Store.
- **ttl** (int) – time to live for the credential, in other words, how long until it expires, expressed in seconds.
- **packages** (Optional[Sequence[*Package*]]) – Sequence of packages to limit the credentials to.
- **channels** (Optional[Sequence[str]]) – Sequence of channel names to limit the credentials to.

Raises *errors.CredentialsAlreadyAvailable* – if credentials already exist.

Return type str

logout()

Clear credentials.

Raises *errors.CredentialsUnavailable* – if credentials cannot be found.

Return type None

notify_revision(*, *name*, *revision_request*)

Post to the revisions endpoint to notify the store about an upload.

This request usually takes place after a successful *upload*.

Parameters

- **name** (str) –
- **revision_request** (*RevisionsRequestModel*) –

Return type *RevisionsResponseModel*

release(*, *name*, *release_request*)
Request a release of name.

Parameters

- **name** (str) – name to release.
- **release_request** (Sequence[*ReleaseRequestModel*]) – sequence of items to release.

Return type None

request(*method*, *url*, *params=None*, *headers=None*, ***kwargs*)
Perform an authenticated request if auth_headers are True.

Parameters

- **method** (str) – HTTP method used for the request.
- **url** (str) – URL to request with method.
- **params** (Optional[Dict[str, str]]) – Query parameters to be sent along with the request.
- **headers** (Optional[Dict[str, str]]) – Headers to be sent along with the request.

Raises

- *errors.StoreServerError* – for error responses.
- *errors.NetworkError* – for lower level network issues.
- *errors.CredentialsUnavailable* – if credentials cannot be found.

Return type Response

Returns Response from the request.

upload_file(*, *filepath*, *monitor_callback=None*)
Upload filepath to storage.

The monitor_callback is a method receiving one argument of type MultipartEncoder, the total length of the upload can be accessed from this encoder from the len attribute to setup a progress bar instance.

The callback is to return a function that receives a MultipartEncoderMonitor from which the .bytes_read attribute can be read to update progress.

The simplest implementation can look like:

```
def monitor_callback(encoder: requests_toolbelt.MultipartEncoder):

    # instantiate progress class with total bytes encoder.len

    def progress_printer(monitor: requests_toolbelt.MultipartEncoderMonitor):
        # Print progress using monitor.bytes_read

    return progress_printer
```

Parameters

- **monitor_callback** (Optional[Callable]) – a callback to monitor progress.
- **filepath** (Path) –

Return type str

whoami()
Return whoami json data queyring *endpoints.Endpoints.whoami*.

Return type Dict[str, Any]

class craft_store.**HTTPClient**(*, *user_agent*)
Bases: object

Generic HTTP Client to communicate with Canonical's Developer Gateway.

This client has a requests like interface, it creates a requests.Session on initialization to handle retries over HTTP and HTTPS requests.

The default number of retries is set in *REQUEST_TOTAL_RETRIES* and can be overridden with the CRAFT_STORE_RETRIES environment variable.

The backoff factor has a default set in *REQUEST_BACKOFF* and can be overridden with the CRAFT_STORE_BACKOFF environment variable.

Retries are done for the following return codes: 500, 502, 503 and 504.

Variables **user_agent** – User-Agent header to identify the client.

Parameters **user_agent** (str) –

get(**args*, ***kwargs*)
Perform an HTTP GET request.

Return type Response

post(**args*, ***kwargs*)
Perform an HTTP POST request.

Return type Response

put(**args*, ***kwargs*)
Perform an HTTP PUT request.

Return type Response

request(*method*, *url*, *params=None*, *headers=None*, ***kwargs*)
Send a request to url.

user_agent is set as part of the headers for the request. All requests are logged through a debug logs, headers matching Authorization and Macaroons have their value replaced.

Parameters

- **method** (str) – HTTP method used for the request.
- **url** (str) – URL to request with method.
- **params** (Optional[Dict[str, str]]) – Query parameters to be sent along with the request.
- **headers** (Optional[Dict[str, str]]) – Headers to be sent along with the request.

Raises

- *errors.StoreServerError* – for error responses.
- *errors.NetworkError* – for lower level network issues.

Return type Response

Returns Response from the request.

class craft_store.**StoreClient**(*, *base_url*, *storage_base_url*, *endpoints*, *application_name*, *user_agent*, *environment_auth=None*, *ephemeral=False*)

Bases: *craft_store.base_client.BaseClient*

Encapsulates API calls for the Snap Store or Charmhub.

Parameters

- **base_url** (str) –
- **storage_base_url** (str) –
- **endpoints** (*Endpoints*) –
- **application_name** (str) –
- **user_agent** (str) –
- **environment_auth** (Optional[str]) –
- **ephemeral** (bool) –

TOKEN_TYPE: str = 'macaroon'

class craft_store.**UbuntuOneStoreClient**(*, *base_url*, *storage_base_url*, *auth_url*, *endpoints*, *application_name*, *user_agent*, *environment_auth=None*, *ephemeral=False*)

Bases: *craft_store.base_client.BaseClient*

Encapsulates API calls for the Snap Store or Charmhub with Ubuntu One.

Parameters

- **base_url** (str) –
- **storage_base_url** (str) –
- **auth_url** (str) –
- **endpoints** (*Endpoints*) –
- **application_name** (str) –
- **user_agent** (str) –
- **environment_auth** (Optional[str]) –
- **ephemeral** (bool) –

TOKEN_TYPE: str = 'u1-macaroon'

request(*method*, *url*, *params=None*, *headers=None*, ***kwargs*)

Perform an authenticated request if auth_headers are True.

Parameters

- **method** (str) – HTTP method used for the request.
- **url** (str) – URL to request with method.
- **params** (Optional[Dict[str, str]]) – Query parameters to be sent along with the request.
- **headers** (Optional[Dict[str, str]]) – Headers to be sent along with the request.

Raises

- *errors.StoreServerError* – for error responses.

- *errors.NetworkError* – for lower level network issues.
- *errors.CredentialsUnavailable* – if credentials cannot be found.

Return type Response

Returns Response from the request.

CHAPTER

FOUR

CHANGELOG

4.1 2.2.1 (2022-08-25)

- Export *craft_store.models.SnapListReleasesModel* and *craft_store.models.CharmListReleasesModel*
- Remove incorrectly exported *SnapChannelMapModel* and *CharmChannelMapModel*
- Make bases optional in *craft_store.models.SnapListReleasesModel*

4.2 2.2.0 (2022-08-11)

- Refactor common code in *endpoints*
- Export new symbols in craft_store.models:
  - `craft_store.models.CharmChannelMapModel`
  - *craft_store.models.MarshableModel*
  - *craft_store.models.ReleaseRequestModel*
  - *craft_store.models.RevisionsRequestModel*
  - *craft_store.models.RevisionsResponseModel*
  - `craft_store.models.SnapChannelMapModel`
- Catch the correct `JSONDecodeError`

4.3 2.1.1 (2022-04-26)

- Update macaroon refresh logic for *craft_store.UbuntuOneStoreClient*

4.4 2.1.0 (2022-03-19)

- Support for ephemeral logins in *craft_store.BaseClient*
- New endpoint to complete the upload experience *craft_store.BaseClient.notify_revision()*
- New endpoint to release *craft_store.BaseClient.release()* and retrieve release information *craft_store.BaseClient.get_list_releases()*
- Support for Python 3.10

4.5 2.0.1 (2022-02-10)

- Convert login expiration to a ISO formatted datetime for Ubuntu endpoints
- Raise *craft_store.errors.CredentialsNotParseable* on base64 decode errors
- Use network location as keyring storage location instead of full base url in *craft_store.base_client.BaseClient*

4.6 2.0.0 (2022-02-07)

- New endpoint for uploads to storage, *craft_store.StoreClient* and *craft_store.UbuntuOneStoreClient* require a new initialization new parameter
- Setting credentials while credentials are already set is no longer allowed *craft_store.errors.CredentialsAlreadyAvailable* is raised if credentials already exist
- NotLoggedIn exception renamed to *craft_store.errors.CredentialsUnavailable*
- Early checks are now in place for keyring availability before a login attempt takes place

4.7 1.2.0 (2021-12-09)

- New whoami endpoint for *craft_store.endpoints.CHARMHUB*
- New class to provide login support for Ubuntu One SSO *craft_store.UbuntuOneStoreClient*

4.8 1.1.0 (2021-11-19)

- Support for channels and packages in endpoints
- *craft_store.store_client.StoreClient* support for retrieving credentials from an environment variable
- Login credentials now returned from *craft_store.BaseClient.login()*

4.9 1.0.0 (2021-10-21)

- Initial release

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX

D

E

G

H

K

L

M

N

P

R

S

T

U

V

W