Camlipy Documentation

Release 0.1.1

Thomas Sileo

October 02, 2013

CONTENTS

1 Quickstart **3**

2 User Guide **5**
2.1 User Guide 5

3 Command-line tool **9**
3.1 Command-line tool 9

4 Development **11**
4.1 Contribution 11

5 API Documentation **13**
5.1 API Documentation 13

Python Module Index **17**

Unofficial Python client for Camlistore.

Release v0.1.1.

Camlipy try to behave exactly the same way that the original Camlistore Go client. It means you can download file uploaded with `camput` or the web ui, and file uploaded with Camlipy works well with the ui and camget.

CHAPTER

ONE

QUICKSTART

```
from camlipy import Camlistore

c = Camlistore('http://localhost:3179')

# Basic put/get
my_blob = 'my blob'
blob_ref = c.put_blob(my_blob)

restored_blob = c.get_blob(blob_ref)

# Retrieve a blob
c.get_blob('sha1-0d31c43041edf303d9d136c918a1337abc9bde97')

# Dump blobs without metadata
c.put_blobs(['My Blob'])
# or
with open('/path/to/file', 'rb') as fh:
    c.put_blobs([fh])

# Put/restore files
c.put_file('/path/to/myfile')
# or
c.put_file(fileobj=open('/path/to/myfile'), permanode=True)

# Get as a fileobj (temporary file)
c.get_file('sha1-0d31c43041edf303d9d136c918a1337abc9bde97')
# Or get directly in a file
with open('/path/to/restored_file', 'wb') as fh:
    c.get_file('sha1-0d31c43041edf303d9d136c918a1337abc9bde97', fh)

# Put/restore directories
blob_ref = c.put_directory('/path/to/my/dir')
c.get_directory(blob_ref)

# Search blobs
c.search('tag:mytag')
```

CHAPTER

TWO

USER GUIDE

2.1 User Guide

2.1.1 Installation

$ sudo pip install camlipy

2.1.2 Getting started

First, you need to create a `Camlistore` before interacting with the API.

```
from camlipy import Camlistore

c = Camlistore('http://localhost:3179')
```

If you have authentication enabled, just provide a tuple `('username', 'password')`.

```
from camlipy import Camlistore

c = Camlistore('http://localhost:3179', auth=('username', 'password'))
```

In the following examples, `c` is always an instance of `Camlistore`.

Each blob is identified by its unique hash, its blob ref, like sha1-bd7d19bf8cf5fdbe955ac17541e215989f2a9ba7.

Raw blobs

Here is how to put/get raw blobs, i.e. without any schema. You can either put a string, or a fileobj like object.

When you call get blob, you get a SpooledTemporaryFile.

```
blob_ref = c.put_blob('my data')

print c.get_blob(blob_ref).read()
```

You can also upload many blobs at once:

```
blob_ref = c.put_blobs(['my data', open('myfile', 'rb')])
```

Files

Uploading files

You can either specify the path:

```
blob_ref = c.put_file('/path/to/file')
```

Or directly a fileobj like object:

```
with open('/path/to/file', 'rb') as fh:
    blob_ref = c.put_file(fileobj=fh)
```

To create a permanode along with the file, just add `permanode=True`, and optionally a list of `tags`.

```
blob_ref = c.put_file('/path/to/file',
                      permanode=True,
                      tags=['list', 'of', 'tags'])
```

Restoring files

`get_file` returns a SpooledTemporaryFile by default.

```
fileobj_res = c.get_file('sha1-bd7d19bf8cf5fdbe955ac17541e215989f2a9ba7')
```

But you can also pass a fileobj directly.

```
with open('/path/to/restored_file', 'wb') as fh:
    fileobj_res = c.get_file('sha1-bd7d19bf8cf5fdbe955ac17541e215989f2a9ba7',
                             fileobj=fh)
```

Directories

Upload directories

Just specify the path:

```
blob_ref = c.put_directory('/path/to/dir')
```

Like when uploading a file, you create a permanode just by passing `permanode=True`, and optionally a list of `tags`.

```
blob_ref = c.put_directory('/path/to/dir',
                           permanode=True,
                           tags=['my tag'])
```

Restore directories

```
c.get_directory('sha1-bd7d19bf8cf5fdbe955ac17541e215989f2a9ba7',
                '/path/to/restored_dir')
```

Exclude files/directories

Camlipy relies on Dirtools to support gitignore like syntax for excluding files/directories, it will looks for a `.exclude` file at the root, check out Dirtools documentation for more informations.

Schema

Schema attribute are stored in a dict under the data attribute. You can retrieve data attribute like standard attribute, i.e. `permanode.data['claimData']` or `permanode.claimData` is the same.

Permanode

You can play directly with the `Permanode` object.

```
# Create a new permanode
permanode = c.permanode()
permanode.save(camli_content, title='My Title', tags=['list', 'of', 'tags'])
# Or load an existing one
permanode = c.permanode(permanode_blob_ref)

# Get/set the camliContent blob ref
blob_ref = permanode.get_camli_content()

permanode.set_camli_content(new_camli_content)

# Also handle camliMember
# Get/set the camliMember blob ref
blob_ref = permanode.get_camli_member()

permanode.add_camli_member(new_camli_member)

# You can also set/get any attribute
permanode.set_attr('title', 'My New Title')
permanode.get_attr('title')

# Fetch the claims history
claims = permanode.claims()

# Fetch a permanode by title
p = c.permanode_by_title('title')
```

Planned permanode

A planned permanode is like a standard permanode except it must have a meaningful `key` and `claim_date`.

```
# Create a new planned permanode
permanode = c.planned_permanode()
permanode.save(camli_content, key='permanode_key', claim_date=datetime(2013, 9, 23, 13, 3, 10))
# Or load an existing one
permanode = c.planned_permanode(permanode_blob_ref)

# Get/set the camliContent blob ref
blob_ref = permanode.get_camli_content()
```

Static set

You can also create static set easily.

```
static_set = c.static_set()
static_set_br = static_set.save([br1, br2, br3])
```

Or you can use the `add_to_static_set` shortcut:

```
static_set_br = c.add_to_static_set([br1, br2, br3])
```

Load an existing static set:

```
static_set = c.static_set(static_set_br)
members = static_set.members
```

You can create a new static while updating its members:

```
static_set = c.static_set(static_set_br)
new_static_set_br = static_set.update([c.put_blob('my new blob')])
```

CHAPTER

THREE

COMMAND-LINE TOOL

3.1 Command-line tool

Camlipy is bundled with a basic command-line tool, `camlipy` that let put/get blobs. It supports raw blob, file and directory.

```
$ camlipy config https://mycamlistorehost.com
$ # or
$ camlipy config https://mycamlistorehost.com user password
$ # Upload
$ camlipy put /path/to/file
$ camlipy put /this/path --permanode
$ echo 'My Blob' | camlipy put -
# Restore
$ camlipy get sha1-0d31c43041edf303d9d136c918a1337abc9bde97
$ camlipy get sha1-0d31c43041edf303d9d136c918a1337abc9bde97 --contents
```

CHAPTER

FOUR

DEVELOPMENT

Tests runs with `devcam server` (Camlistore development server), see HACKING from Camlistore.

To execute the tests, just run:

```
$ python setup.py test
```

4.1 Building the C extension

You must have swig installed.

$ python setup.py build_ext –inplace $ cd camlipy $ swig -python rollsum.i

4.2 Contribution

Feel free to submit a pull request!

CHAPTER

FIVE

API DOCUMENTATION

5.1 API Documentation

5.1.1 camlipy

Camlistore client.

camlipy.**compute_hash**(*data*, *blocksize=4096*)
Return the hash object for the file ‘filepath’, processing the file by chunk of ‘blocksize’.

Parameters

- **filepath** (*data*) – string or fileobj
- **blocksize** (*int*) – Size of the chunk when processing the file

camlipy.**check_hash**(*_hash*)
Check if the hash is valid.

class camlipy.**Camlistore**(*server='http://localhost:3179'*, *auth=None*)
Camlistore Python client

Args: server: server address auth: tuple (user, password) if authentication is enabled.

add_to_static_set(*blob_refs*)
Shortcut to create a new static-set.

describe_blob(*blobref*)
Return blob meta data.

get_blob(*blobref*)
Retrieve blob content, return a fileobj. If the blob is a schema, it returns a dict.

get_directory(*br*, *path*)
Shortcut for restoring/retrieving a directory.

Call camlipy.directory.get_directory under the hood.

get_file(*blob_ref*, *fileobj=None*)
Shortcut for downloading/restoring a file.

Call camlipy.filereader.get_file under the hood.

permanode(*blob_ref=None*)
Shortcut to initialize a permanode.

permanode_by_title(*title*, *create=False*)
Shortcut to fetch the first permanode with the given title.

planned_permanode(*blob_ref=None*)
Shortcut to initialize a planned permanode.

put_blob(*blob*)
Shortcut/helper for uploading a single blob.

Blob can be either a string or a fileobj.

put_blobs(*blobs*)
Upload blobs using with standard multi-part upload. Returns a dict with received (blobref and size) and skipped (blobref only)

put_directory(*path*, *permanode=False*)
Shortcut for upload an entire directory.

Call camlipy.directory.put_directory under the hood.

put_file(*path=None*, *fileobj=None*, *permanode=False*)
Shortcut for uploading a file along with its meta-data.

Call camlipy.filewriter.put_file under the hood.

search(*value*, *attr=''*, *fuzzy=False*, *max=100*)
Perform query with the same syntax as Camistore ui.

Examples of queries: - tag:mytag - title:my_text_file.txt - my query

static_set(*blob_ref=None*)
Shortcut to initialize a static-set.

5.1.2 camlipy.directory

Helper for uploading recursively directory.

camlipy.directory.**get_directory**(*con*, *br*, *path*)
Download a directory.

camlipy.directory.**put_directory**(*con*, *path*, *permanode=False*)
Helper to put a directory.

5.1.3 camlipy.filereader

Read the file schema, and output chunk in a temp file.

camlipy.filereader.**get_file**(*con*, *blob_ref*, *fileobj=None*)
Helper for download a file from his blobRef to a fileobj.

5.1.4 camlipy.filewriter

Helper for uploading file, takes care of chunking file, create the file schema.

class camlipy.filewriter.**Span**(*_from=0*, *to=0*, *bits=None*, *children=[]*, *chunk_cnt=0*, *br=None*, *size=None*)
Chunk metadata, used to create the tree, and compute chunk/bytesRef size.

camlipy.filewriter.**put_file**(*con*, *path=None*, *fileobj=None*, *permanode=False*)
Helper for uploading a file to a Camlistore server.

Specify either a path, or a fileobj.

Can also create a permanode.

5.1.5 camlipy.schema

Helper for creating/loading schemas.

class camlipy.schema.**Bytes**(*con*, *br=None*)
Bytes schema.

class camlipy.schema.**Claim**(*con*, *permanode_blobref*, *claim_blobref=None*)
Claim schema with support for set/add/del attribute.

class camlipy.schema.**Directory**(*con*, *path=None*, *blob_ref=None*)
Directory Schema

class camlipy.schema.**File**(*con*, *path=None*, *file_name=None*, *blob_ref=None*)
File schema.

file_name is guessed from path if provided.

class camlipy.schema.**FileCommon**(*con*, *path=None*, *blob_ref=None*)
FileCommon schema.

class camlipy.schema.**Permanode**(*con*, *permanode_blob_ref=None*)
Permanode Schema with helpers for claims.

add_attr(*attr*, *value*)
Create a claim to add avlue to attr.

add_camli_member(*camli_member*)
Append a new camliMember.

claims()
Return claims for the current permanode.

delete_attr(*attr*, *value=None*)
Create a claim to delete attr/attr:value.

get_attr(*attr*)
Retrieve attr from indexer.

get_camli_content()
Fetch the current camliContent blobRef.

get_camli_member()
Fetch the current camliMember blobRef.

save(*camli_content=None*, *camli_member=None*, *title=None*, *tags=[]*)
Create the permanode, takes optional title and tags.

set_attr(*attr*, *value*, *claim_date=None*)
Create a claim to set attr to value.

set_camli_content(*camli_content*, *claim_date=None*)
Create a new camliContent claim.

class camlipy.schema.**PlannedPermanode**(*con*, *permanode_blob_ref=None*)
A planned permanode is like a normal permanode, except it have a meaningful "key" key, and a meaningful "claimDate", so the signature for the given key, claimDate is always the same.

class camlipy.schema.**Schema**(*con*, *blob_ref=None*)
Basic Schema base class.

Also used to load (and decoding?) existing schema.

Args: con: Camlistore instance blob_ref: Optional blobRef if the blob already exists.

describe()
Call the API to describe the blob.

json()
Return json data.

sign()
Return signed json.

class camlipy.schema.**StaticSet**(*con*, *br=None*)
StaticSet schema.

update(*members=[]*)
Update a static-set by creating a new one.

camlipy.schema.**dt_to_camli_iso**(*dt*)
Convert a datetime to iso datetime compatible with camlistore.

camlipy.schema.**get_stat_info**(*path*)
Return OS stat info for the given path.

camlipy.schema.**ts_to_camli_iso**(*ts*)
Convert timestamp to UTC iso datetime compatible with camlistore.

5.1.6 camlipy.search

Search API wrapper.

class camlipy.search.**Search**(*con*)
Basic search wrapper around the API.

search(*value*, *attr=''*, *fuzzy=False*, *max=100*)
Perform query with the same syntax as Camistore ui.

Examples of queries: - tag:mytag - title:my_text_file.txt - my query

PYTHON MODULE INDEX

C

camlipy, 13
camlipy.directory, 14
camlipy.filereader, 14
camlipy.filewriter, 14
camlipy.schema, 15
camlipy.search, 16