hs_restclient Documentation

Release 1.3.6

HydroShare

Mar 25, 2020

Contents

Release v1.3.6

CHAPTER 1

Installation

pip install hs_restclient

CHAPTER 2

API documentation

modindex

CHAPTER 3

Usage

To get system metadata for public resources:

```
>>> from hs_restclient import HydroShare
>>> hs = HydroShare()
>>> for resource in hs.resources():
>>>     print(resource)
```

To authenticate using HTTP Basic authentication, and then get system metadata for resources you have access to:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> for resource in hs.resources():
>>>     print(resource)
```

To authenticate using HTTP Basic authentication, with input prompt for username and password:

```
>>> from hs_restclient import HydroShare
>>> hs = HydroShare()
>>> for resource in hs.resources():
>>>     print(resource)
```

To authenticate using OAuth2 authentication (using a user and password supplied by the user), and then get a list of resources you have access to:

```
>>> from oauthlib.oauth2 import TokenExpiredError
>>> from hs_restclient import HydroShare, HydroShareAuthOAuth2
>>>
>>> # Get a client ID and client secret by registering a new application at:
>>> # https://www.hydroshare.org/o/applications/
>>> # Choose client type "Confidential" and authorization grant type "Resource owner␣
↪password-based"
>>> # Keep these secret!
>>> client_id = 'MYCLIENTID'
```

(continues on next page)

(continued from previous page)

```
>>> client_secret = 'MYCLIENTSECRET'
>>>
>>> auth = HydroShareAuthOAuth2(client_id, client_secret,
>>>                             username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>>
>>> try:
>>>     for resource in hs.resources():
>>>         print(resource)
>>> except TokenExpiredError as e:
>>>     hs = HydroShare(auth=auth)
>>>    for resource in hs.resources():
>>>         print(resource)
```

Note that currently the client does not handle token renewal, hence the need to catch TokenExpiredError.

To authenticate using OAuth2 authentication (using an existing token), and then get a list of resources you have access to:

```
>>> from oauthlib.oauth2 import TokenExpiredError
>>> from hs_restclient import HydroShare, HydroShareAuthOAuth2
>>>
>>> # Get a client ID and client secret by registering a new application at:
>>> # https://www.hydroshare.org/o/applications/
>>> # Choose client type "Confidential" and authorization grant type "Resource owner␣
↪password-based"
>>> # Keep these secret!
>>> client_id = 'MYCLIENTID'
>>> client_secret = 'MYCLIENTSECRET'
>>>
>>> # A token dictionary obtained separately from HydroShare of the form:
>>> #   {
>>> #       "access_token": "<your_access_token>",
>>> #       "token_type": "Bearer",
>>> #       "expires_in": 36000,
>>> #       "refresh_token": "<your_refresh_token>",
>>> #       "scope": "read write groups"
>>> #   }
>>>
>>> # get_token() is a stand in for how you get a new token on your system.
>>> token = get_token()
>>> auth = HydroShareAuthOAuth2(client_id, client_secret,
>>>                             token=token)
>>> try:
>>>     hs = HydroShare(auth=auth)
>>>     for resource in hs.resources():
>>>         print(resource)
>>> except:
>>>     # get_token() is a stand in for how you get a new token on your system.
>>>     token = get_token()
>>>     auth = HydroShareAuthOAuth2(client_id, client_secret,
>>>                                 token=token)
>>>     hs = HydroShare(auth=auth)
>>>     for resource in hs.resources():
>>>         print(resource)
```

Note that currently the client does not handle token renewal, hence the need to catch TokenExpiredError.

To connect to a development HydroShare server that uses a self-sign security certificate:

```
>>> from hs_restclient import HydroShare
>>> hs = HydroShare(hostname='mydev.mydomain.net', verify=False)
>>> for resource in hs.resources():
>>>     print(resource)
```

To connect to a development HydroShare server that is not running HTTPS:

```
>>> from hs_restclient import HydroShare
>>> hs = HydroShare(hostname='mydev.mydomain.net', port=8000, use_https=False)
>>> for resource in hs.resources():
>>>     print(resource)
```

Note that authenticated connections **must** use HTTPS.

To get the system metadata for a particular resource:

```
>>> from hs_restclient import HydroShare
>>> hs = HydroShare()
>>> resource_md = hs.getSystemMetadata('e62a438bec384087b6c00ddcd1b6475a')
>>> print(resource_md['resource_title'])
```

To get the BagIt archive of a particular resource:

```
>>> from hs_restclient import HydroShare
>>> hs = HydroShare()
>>> hs.getResource('e62a438bec384087b6c00ddcd1b6475a', destination='/tmp')
```

or to have the BagIt archive unzipped for you:

```
>>> from hs_restclient import HydroShare
>>> hs = HydroShare()
>>> hs.getResource('e62a438bec384087b6c00ddcd1b6475a', destination='/tmp', unzip=True)
```

or to get the BagIt archive as a generator (sort of like a buffered stream):

```
>>> from hs_restclient import HydroShare
>>> hs = HydroShare()
>>> resource = hs.getResource('e62a438bec384087b6c00ddcd1b6475a')
>>> with open('/tmp/myresource.zip', 'wb') as fd:
>>>     for chunk in resource:
>>>         fd.write(chunk)
```

Note that when the BagIt archive is not ready for download (this archive zip file needs to be recreated) the client by default will wait until the BagIt archive is recreated. This may take a while. If you want to get the BagIt archive only if it is ready for download:

```
>>> from hs_restclient import HydroShare
>>> hs = HydroShare()
>>>try:
>>>     hs.getResource('e62a438bec384087b6c00ddcd1b6475a', destination='/tmp', wait_
↪for_bag_creation=False)
>>>except HydroShareBagNotReadyException as e:
>>>     print('BagIt file is being generated and not ready for download at this time.
↪')
```

To create a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> abstract = 'My abstract'
>>> title = 'My resource'
>>> keywords = ('my keyword 1', 'my keyword 2')
>>> rtype = 'GenericResource'
>>> fpath = '/path/to/a/file'
>>> metadata = '[{"coverage":{"type":"period", "value":{"start":"01/01/2000", "end":
↪"12/12/2010"}}}, {"creator":{"name":"John Smith"}}, {"creator":{"name":"Lisa Miller
↪"}}]'
>>> extra_metadata = '{"key-1": "value-1", "key-2": "value-2"}'
>>> resource_id = hs.createResource(rtype, title, resource_file=fpath,␣
↪keywords=keywords, abstract=abstract, metadata=metadata, extra_metadata=extra_
↪metadata)
```

To make a resource public:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> hs.setAccessRules('ID OF RESOURCE GOES HERE', public=True)
```

To delete a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> hs.deleteResource('ID OF RESOURCE GOES HERE')
```

To add a file to a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> fpath = '/path/to/somefile.txt'
>>> resource_id = hs.addResourceFile('ID OF RESOURCE GOES HERE', fpath)
```

To get a file in a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> fname = 'somefile.txt'
>>> fpath = hs.getResourceFile('ID OF RESOURCE GOES HERE', fname, destination='/
↪directory/to/download/file/to')
```

To delete a file from a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> fname = 'somefile.txt'
>>> resource_id = hs.deleteResourceFile('ID OF RESOURCE GOES HERE', fname)
```

To get resource map xml data for a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> resource_map_xml = hs.getResourceMap('ID OF RESOURCE GOES HERE')
```

To get the contents of a specific folder of a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> folder_name = 'some_folder'
>>> folder_contents_json = hs.getResourceFolderContents('ID OF RESOURCE GOES HERE',␣
↪pathname=folder_name)
```

To create a folder for a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> folder_to_create = "folder_1/folder_2"
>>> response_json = hs.createResourceFolder('ID OF RESOURCE GOES HERE',␣
↪pathname=folder_to_create)
```

To delete a folder for a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> folder_to_delete = "folder_1/folder_2"
>>> response_json = hs.deleteResourceFolder('ID OF RESOURCE GOES HERE',␣
↪pathname=folder_to_delete)
```

To get science metadata as xml+rdf data for a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> science_metadata_xml = hs.getScienceMetadataRDF('ID OF RESOURCE GOES HERE')
```

To get science metadata as json data (Dublin core metadata only) for a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> science_metadata_json = hs.getScienceMetadata('ID OF RESOURCE GOES HERE')
```

To update science metadata (Dublin core metadata only) for a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> metadata = {
                "title": "A new title for my resource",
                "coverages": [
                                                  {"type": "period", "value
↪": {"start": "01/01/2000", "end": "12/12/2010"}}
                                               ],
```

(continues on next page)

(continued from previous page)

```
                              "creators": [
                                                           {"name": "John Smith",
↪"organization": "USU"},
                                                           {"name": "Lisa Miller", "email
↪": "lisa_miller@gmail.com"}
                                                    ]
                           }
>>> science_metadata_json = hs.updateScienceMetadata('ID OF RESOURCE GOES HERE',␣
↪metadata=metadata)
```

To update custom science metadata (non-Dublin core) for a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> metadata = {
                 "weather": "sunny",
                 "temp": "80C"
                          }
>>> result = hs.resource('ID OF RESOURCE GOES HERE').scimeta.custom(metadata)
```

To move or rename a resource file:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> options = {
                 "source_path": "/source/path/file.txt",
                 "target_path": "/target/path/file.txt"
                          }
>>> result = hs.resource('ID OF RESOURCE GOES HERE').functions.move_or_rename(options)
```

To zip a resource file or folder:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> options = {
                 "input_coll_path": "/source/path/file.txt",
                 "target_path": "/target/path/file.txt",
                 "remove_original_after_zip": True
                          }
>>> result = hs.resource('ID OF RESOURCE GOES HERE').functions.zip(options)
```

To unzip a resource file or folder:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> options = {
                 "zip_with_rel_path": "/source/path/file.zip",
                 "remove_original_zip": True,
                 "overwrite": False
                          }
>>> result = hs.resource('ID OF RESOURCE GOES HERE').functions.unzip(options)
```

To create a copy of a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> result = hs.resource('ID OF RESOURCE GOES HERE').copy()
```

To create a new version of a resource:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> result = hs.resource('ID OF RESOURCE GOES HERE').version()
```

To upload files to a specific resource folder:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> local_file = "file_name.txt"
>>> resource_filename = "folder_name/file_name.txt"
>>> result = hs.addResourceFile('ID OF RESOURCE GOES HERE', local_file, resource_
↪filename)
```

To create a referenced content file:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> name = "file_name"
>>> ref_url = "https://www.hydroshare.org"
>>> path = "data/contents"
>>> response_json = hs.createReferenceURL('ID OF RESOURCE GOES HERE', name, ref_url,␣
↪path)
```

To update a referenced content file:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> path = "data/contents"
>>> name = "file_name"
>>> ref_url = "https://www.cuahsi.org"
>>> response_json = hs.updateReferencedFile('ID OF RESOURCE GOES HERE', path, name,␣
↪ref_url)
```

To set resource flags:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> options = {
                 "flag": "one of make_public, make_private, make_shareable,
        make_not_shareable, make_discoverable, make_not_discoverable"
              }
>>> result = hs.resource('ID OF RESOURCE GOES HERE').flag(options)
```

Alternatively, you can use the helper functions:

```
>>>  hs.resource('ID OF RESOURCE GOES HERE').public(True) # or False
>>>  hs.resource('ID OF RESOURCE GOES HERE').discoverable(True) # or False
>>>  hs.resource('ID OF RESOURCE GOES HERE').shareable(True) # or False
```

To discover resources via subject or bounding box:

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> result = hs.resources(subject="comma,separated,list,of,subjects")
```

```
>>> result = hs.resources(coverage_type="box",
                          north="50",
                          south="30",
                          east="40",
                          west="20")
```

To discover resources via other parameters

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
```

```
>>> # Discover via creator, group, user, owner
>>> resources = hs.resources(creator="email or username")
>>> resources = hs.resources(user="email or username")
>>> resources = hs.resources(owner="email or username")
>>> resources = hs.resources(author="email or username")
>>> resources = hs.resources(group="id or name")
```

from_date allows you to specify the earliest creation date to query for. Use any python datetime object i.e. datetime.datetime(2018, 09, 07)

to_date allows you to specify the latest creation date to query for Use any python datetime object i.e. datetime.datetime(2018, 09, 07)

```
>>> # Discover via date range (datetime objects)
>>> resources = hs.resources(from_date=datetime, to_date=datetime)
```

start allows you to specify the index of the resource to start querying from

count allows you to specify how many resources to include in the query results

```
>>> # Discover via start or count (integers)
>>> resources = hs.resources(start=4)
>>> resources = hs.resources(count=4)
```

```
>>> # Discover via full text search
>>> resources = hs.resources(full_text_search="any text here")
```

```
>>> # Discover via flags (boolean)
>>> resources = hs.resources(published=False)
>>> resources = hs.resources(edit_permission=False)
>>> resources = hs.resources(public=False)
>>> # Discover via resource type
>>> resources = hs.resources(type=None)
```

To get a list of all resource files

```
>>> # List all resource files
>>> hs.resource('ID OF RESOURCE GOES HERE').files.all().content
```

To get file metadata

```
>>> hs.resource('ID OF RESOURCE GOES HERE').files.metadata(pathname).content
```

To set file metadata

```
>>> # This is a PUT request so the entire object, all parameters are overwritten
>>> params = {}
>>> params['keywords'] = ["keyword1","keyword2"]
>>> params['spatial_coverage'] = {
        "units":"Decimal degrees",
        "east":-90.0465,
        "north":48.6791,
        "name":"12232",
        "projection":"WGS 84 EPSG:4326"
    }
>>> params['temporal_coverage'] = {"start":"2018-02-23","end":"2018-02-29"}
>>> params['extra_metadata'] = {"extended1":"one"}
>>> params['title'] = "New Metadata Title"
>>> hs.resource('ID OF RESOURCE GOES HERE').files.metadata(pathname, params)
```

To set a file to a file type (e.g., NetCDF) in a composite resource: Note: Allowed file type are: NetCDF, GeoRaster and GeoFeature

```
>>> from hs_restclient import HydroShare, HydroShareAuthBasic
>>> auth = HydroShareAuthBasic(username='myusername', password='mypassword')
>>> hs = HydroShare(auth=auth)
>>> options = {
                 "file_path": "file.nc",
                 "hs_file_type": "NetCDF"
                          }
>>> result = hs.resource('ID OF RESOURCE GOES HERE').functions.set_file_type(options)
```

CHAPTER 4

Index

- genindex