ChemSpiPy Documentation

Release 1.0.1

Matt Swain

Jul 09, 2018

Contents

ChemSpiPy provides a way to interact with ChemSpider in Python. It allows chemical searches, chemical file downloads, depiction and retrieval of chemical properties. Here's a quick peek:

```
>>> from chemspipy import ChemSpider
>>> cs = ChemSpider('<YOUR-SECURITY-TOKEN>')
>>> c1 = cs.get_compound(236)  # Specify compound by ChemSpider ID
>>> c2 = cs.search('benzene')  # Search using name, SMILES, InChI,␣
↪InChIKey, etc.
```

CHAPTER 1

Features

- Search compounds by synonym, SMILES, InChI, InChIKey, formula and mass.
- Get identifiers and calculated properties for any compound record in ChemSpider.
- Download compound records as a MOL file with 2D or 3D coordinates.
- Get a 2D compound depiction as a PNG image.
- Retrieve all available spectral information for a specific compound.
- Complete interface to every endpoint of the ChemSpider Web APIs.
- Supports Python versions 2.7 – 3.4.

CHAPTER 2

User guide

A step-by-step guide to getting started with ChemSpiPy.

2.1 Introduction

ChemSpiPy is a Python wrapper that allows simple access to the web APIs offered by ChemSpider. The aim is to provide an interface for users to access and query the ChemSpider database using Python, facilitating programs that can automatically carry out the tasks that you might otherwise perform manually via the ChemSpider website (http://www.chemspider.com).

The ChemSpider website has full documentation for the ChemSpider APIs (http://www.chemspider.com/AboutServices.aspx). It can be useful to browse through this documentation before getting started with ChemSpiPy to get an idea of what sort of features are available.

2.1.1 Obtaining a security token

Access to the ChemSpider API is free to academic users. Commercial users should contact the ChemSpider team to obtain access.

Most operations require a "security token" that is issued to you automatically when you register for a RSC ID (https://www.rsc.org/rsc-id/sign-in) and then sign in to ChemSpider. Once you have done this, you can find your security token on your ChemSpider User Profile (http://www.chemspider.com/UserProfile.aspx).

Some operations require a further "Service Subscriber" role. Contact the ChemSpider team to discuss upgrading your user account for access to these features.

2.1.2 ChemSpiPy license

2.2 Installation

ChemSpiPy supports Python versions 2.7, 3.2, 3.3 and 3.4.

There are two required dependencies: six (http://pythonhosted.org/six/) and requests (http://docs.python-requests.org/).

2.2.1 Option 1: Use pip (recommended)

The easiest and recommended way to install is using pip:

```
pip install chemspipy
```

This will download the latest version of ChemSpiPy, and place it in your *site-packages* folder so it is automatically available to all your python scripts. It should also ensure that the dependencies six (http://pythonhosted.org/six/) and requests (http://docs.python-requests.org/) are installed.

If you don't already have pip installed, you can install it using get-pip.py (http://www.pip-installer.org/en/latest/installing.html):

```
curl -O https://raw.github.com/pypa/pip/master/contrib/get-pip.py
python get-pip.py
```

2.2.2 Option 2: Download the latest release

Alternatively, download the latest release (https://github.com/mcs07/ChemSpiPy/releases/) manually and install yourself:

```
tar -xzvf ChemSpiPy-1.0.1.tar.gz
cd ChemSpiPy-1.0.1
python setup.py install
```

The setup.py command will install ChemSpiPy in your *site-packages* folder so it is automatically available to all your python scripts.

2.2.3 Option 3: Clone the repository

The latest development version of ChemSpiPy is always available on GitHub (https://github.com/mcs07/ChemSpiPy). This version is not guaranteed to be stable, but may include new features that have not yet been released. Simply clone the repository and install as usual:

```
git clone https://github.com/mcs07/ChemSpiPy.git
cd ChemSpiPy
python setup.py install
```

2.3 Getting started

This page gives a introduction on how to get started with ChemSpiPy.

2.3.1 Before we start

- Make sure you have *installed ChemSpiPy* (page 6).
- *Obtain a security token* (page 5) from the ChemSpider web site.

2.3.2 First steps

Start by importing ChemSpider:

```
>>> from chemspipy import ChemSpider
```

Then connect to ChemSpider by creating a `ChemSpider` instance using your security token:

```
>>> cs = ChemSpider('<YOUR-SECURITY-TOKEN>')
```

All your interaction with the ChemSpider database should now happen through this ChemSpider object.

2.3.3 Retrieve a Compound

Retrieving information about a specific Compound in the ChemSpider database is simple.

Let's get the Compound with ChemSpider ID 2157 (http://www.chemspider.com/Chemical-Structure.2157.html):

```
>>> c = cs.get_compound(2157)
```

Now we have a *Compound* (page 18) object called `c`. We can get various identifiers and calculated properties from this object:

```
>>> print(c.molecular_formula)
C_{9}H_{8}O_{4}
>>> print(c.molecular_weight)
180.15742
>>> print(c.smiles)
CC(=O)OC1=CC=CC=C1C(=O)O
>>> print(c.common_name)
Aspirin
```

2.3.4 Search for a name

What if you don't know the ChemSpider ID of the Compound you want? Instead use the `search` method:

```
>>> for result in cs.search('Glucose'):
...     print(result)
Compound(5589)
Compound(58238)
Compound(71358)
Compound(96749)
Compound(9312824)
Compound(9484839)
```

The `search` method accepts any identifer that ChemSpider can interpret, including names, registry numbers, SMILES and InChI.

That's a quick taster of the basic ChemSpiPy functionality. Read on for more some more advanced usage examples.

2.4 Searching

TODO

- **The main search method**
  - Explain how search is done in the background

  - Check if results are ready - get message, duration, count
  - Iterate results just like a python list
  - TODO: Ordered results (csid, mass_defect, molecular_weight, reference_count, datasource_count, pubmed_count, rsc_count)

- search_by_formula
- search_by_mass
- simple_search (?)

2.5 Compound

TODO

- Many methods return Compound objects
- This is a simple wrapper around a ChemSpider ID that allows further information to be retrieved
- Once retrieved, properties are cached so subsequent access on the same Compound object should be faster.
- **Behind the scenes, Compound objects just use other API endpoints:**
  - get_extended_compound_info
  - get_record_mol
  - get_compound_thumbnail

2.6 Advanced

2.6.1 Keep your security token secret

Be careful not to include your security token when sharing code. A simple way to ensure this doesn't happen by accident is to store your security token as an environment variable that can be specified in your *.bash_profile* or *.zshrc* file:

```
export CHEMSPIDER_SECURITY_TOKEN=<YOUR-SECURITY-TOKEN>
```

This can then be retrieved in your scripts using `os.environ`:

```
>>> CST = os.environ['CHEMSPIDER_SECURITY_TOKEN']
>>> cs = ChemSpider(security_token=CST)
```

2.6.2 Specify a User Agent

As well as using your security token, it is possible to identify your program to the ChemSpider servers using a User Agent string.

You can specify a custom User Agent through ChemSpiPy through the optional `user_agent` parameter to the ChemSpider class:

```
>>> from chemspipy import ChemSpider
>>> cs = ChemSpider('<YOUR-SECURITY-TOKEN>', user_agent='My program␣
↪1.3, ChemSpiPy 1.0.1, Python 2.7')
```

2.6.3 Logging

ChemSpiPy can generate logging statements if required. Just set the desired logging level:

```
import logging
logging.basicConfig(level=logging.DEBUG)
```

The logger is named 'chemspipy'. There is more information on logging in the Python logging documentation (http://docs.python.org/2/howto/logging.html).

2.7 Contributing

Contributions of any kind are greatly appreciated!

2.7.1 Feedback

The Issue Tracker (https://github.com/mcs07/ChemSpiPy/issues) is the best place to post any feature ideas, requests and bug reports.

2.7.2 Contributing

If you are able to contribute changes yourself, just fork the source code (https://github.com/mcs07/ChemSpiPy) on GitHub, make changes and file a pull request. All contributions are welcome, no matter how big or small.

Quick guide to contributing

1. Fork the ChemSpiPy repository on GitHub (https://github.com/mcs07/ChemSpiPy/fork), then clone your fork to your local machine:

   ```
   git clone https://github.com/<username>/ChemSpiPy.git
   ```

2. Install the development requirements:

```
cd chemspipy
pip install -r requirements/development.txt
```

3. Create a new branch for your changes:

```
git checkout -b <name-for-changes>
```

4. Make your changes or additions. Ideally add some tests and ensure they pass by running:

```
nosetests
```

   The final line of the output should be `OK`.

5. Commit your changes and push to your fork on GitHub:

```
git add .
git commit -m "<description-of-changes>"
git push origin <name-for-changes>
```

4. Submit a pull request (https://github.com/mcs07/ChemSpiPy/compare/).

Tips

- Follow the PEP8 (https://www.python.org/dev/peps/pep-0008) style guide.
- Include docstrings as described in PEP257 (https://www.python.org/dev/peps/pep-0257).
- Try and include tests that cover your changes.
- Try to write good commit messages (http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html).
- Consider squashing your commits (http://gitready.com/advanced/2009/02/10/squashing-commits-with-rebase.html) with rebase.
- Read the GitHub help page on Using pull requests (https://help.github.com/articles/using-pull-requests).

CHAPTER 3

API documentation

Comprehensive API documentation with information on every function, class and method.

3.1 API documentation

This part of the documentation is automatically generated from the ChemSpiPy source code and comments.

3.1.1 chemspipy.api

Core API for interacting with ChemSpider web services.

class chemspipy.**ChemSpider**

Provides access to the ChemSpider API.

See `BaseChemSpider` further information.

Parameters

- **security_token** (*string*) – (Optional) Your ChemSpider security token.
- **user_agent** (*string*) – (Optional) Identify your application to ChemSpider servers.

async_simple_search(*query*)

Search ChemSpider with arbitrary query, returning results in order of the best match found.

This method returns a transaction ID which can be used with other methods to get search status and results.

Security token is required.

Parameters **query** (*string*) – Search query - a name, SMILES, InChI, InChIKey, CSID, etc.

Returns Transaction ID.

Return type string

async_simple_search_ordered(*query*, *order=u'csid'*, *direction=u'ascending'*)

Search ChemSpider with arbitrary query, returning results with a custom order.

This method returns a transaction ID which can be used with other methods to get search status and results.

Security token is required.

Parameters

- **query** (*string*) – Search query - a name, SMILES, InChI, InChIKey, CSID, etc.
- **order** (*string*) – 'csid', 'mass_defect', 'molecular_weight', 'reference_count', 'datasource_count', 'pubmed_count' or 'rsc_count'.
- **direction** (*string*) – 'ascending' or 'descending'.

Returns Transaction ID.

Return type string

construct_api_url(*api*, *endpoint*, ***params*)

Construct a Chemspider API url, encoded, with parameters as a GET querystring.

Parameters

- **api** (*string*) – The specific ChemSpider API to call (MassSpec, Search, Spectra, InChI).
- **endpoint** (*string*) – ChemSpider API endpoint.
- **params** – (optional) Parameters for the ChemSpider endpoint as keyword arguments.

Return type string

get_async_search_result(*rid*)

Get the results from a asynchronous search operation. Security token is required.

Parameters **rid** (*string*) – A transaction ID, returned by an asynchronous search method.

Returns A list of Compounds.

Return type list (https://docs.python.org/3/library/stdtypes.html#list)[*Compound* (page 18)]

get_async_search_result_part(*rid*, *start=0*, *count=-1*)
Get a slice of the results from a asynchronous search operation. Security token is required.

Parameters

- **rid** (*string*) – A transaction ID, returned by an asynchronous search method.
- **start** (*int* (https://docs.python.org/3/library/functions.html#int)) – The number of results to skip.
- **count** (*int* (https://docs.python.org/3/library/functions.html#int)) – The number of results to return. -1 returns all through to end.

Returns A list of Compounds.

Return type list (https://docs.python.org/3/library/stdtypes.html#list)[*Compound* (page 18)]

get_async_search_status(*rid*)
Check the status of an asynchronous search operation.

Security token is required.

Parameters **rid** (*string*) – A transaction ID, returned by an asynchronous search method.

Returns Unknown, Created, Scheduled, Processing, Suspended, PartialResultReady, ResultReady, Failed, TooManyRecords

Return type string

get_async_search_status_and_count(*rid*)
Check the status of an asynchronous search operation. If ready, a count and message are also returned.

Security token is required.

Parameters **rid** (*string*) – A transaction ID, returned by an asynchronous search method.

Return type dict (https://docs.python.org/3/library/stdtypes.html#dict)

get_compound(*csid*)
Return a Compound object for a given ChemSpider ID. Security token is required.

Parameters **csid** (*string*|*int*) – ChemSpider ID.

Returns The Compound with the specified ChemSpider ID.

Return type *Compound* (page 18)

get_compound_info(*csid*)
Get SMILES, StdInChI and StdInChIKey for a given CSID. Security token is required.

Parameters **csid** (*string*|*int*) – ChemSpider ID.

Return type dict (https://docs.python.org/3/library/stdtypes.html#dict)

get_compound_thumbnail(*csid*)
Get PNG image as binary data.

Parameters **csid** (*string|int*) – ChemSpider ID.

Return type bytes (https://docs.python.org/3/library/stdtypes.html#bytes)

get_compounds(*csids*)
Return a list of Compound objects, given a list ChemSpider IDs. Security token is required.

Parameters **csids** (*list* (https://docs.python.org/3/library/stdtypes.html#list) *[string|* – List of ChemSpider IDs.

Returns List of Compounds with the specified ChemSpider IDs.

Return type list (https://docs.python.org/3/library/stdtypes.html#list)[*Compound* (page 18)]

get_databases()
Get the list of datasources in ChemSpider.

get_extended_compound_info(*csid*)
Get extended record details for a CSID. Security token is required.

Parameters **csid** (*string|int*) – ChemSpider ID.

get_extended_compound_info_list(*csids*)
Get extended record details for a list of CSIDs. Security token is required.

Parameters **csids** (*list* (https://docs.python.org/3/library/stdtypes.html#list) *[string|* – ChemSpider IDs.

get_extended_mol_compound_info_list(*csids*, *mol_type=u'e2D'*, *include_reference_counts=False*, *include_external_references=False*)
Get extended record details (including MOL) for a list of CSIDs.

A maximum of 250 CSIDs can be fetched per request. Security token is required.

Parameters

- **csids** (*list* (https://docs.python.org/3/library/stdtypes.html#list) *[string|int]*) – ChemSpider IDs.
- **mol_type** (*string*) – 2d, 3d or both.
- **include_reference_counts** (*bool* (https://docs.python.org/3/library/functions.html#bool)) – Whether to include reference counts.
- **include_external_references** (*bool* (https://docs.python.org/3/library/functions.html#bool)) – Whether to include external references.

get_original_mol(*csid*)
Get original submitted MOL file. Security token is required.

Parameters **csid** (*string|int*) – ChemSpider ID.

get_record_mol(*csid*, *calc3d=False*)
Get ChemSpider record in MOL format. Security token is required.

Parameters

- **csid** (*string|int*) – ChemSpider ID.
- **calc3d** (*bool* (https://docs.python.org/3/library/functions.html#bool)) – Whether 3D coordinates should be calculated before returning record data.

get_spectra_info_list(*csids*)
Get information about all the spectra for a list of CSIDs.

Parameters **csids** (*list* (https://docs.python.org/3/library/stdtypes.html#list) *[string|* – ChemSpider IDs.

Returns List of spectrum info.

Return type list (https://docs.python.org/3/library/stdtypes.html#list)[dict (https://docs.python.org/3/library/stdtypes.html#dict)]

request(*api*, *endpoint*, ***params*)
Construct API request and return the XML response.

Parameters

- **api** (*string*) – The specific ChemSpider API to call (MassSpec, Search, Spectra, InChI).
- **endpoint** (*string*) – ChemSpider API endpoint.
- **params** – (optional) Parameters for the ChemSpider endpoint as keyword arguments.

Return type xml tree

search(*query*)
Search ChemSpider for the specified query and return the results. Security token is required.

search_by_formula(*formula*)
Search ChemSpider by molecular formula.

Parameters **formula** (*string*) – Molecular formula

Returns A list of Compounds.

Return type list (https://docs.python.org/3/library/stdtypes.html#list)[*Compound* (page 18)]

search_by_mass(*mass*, *mass_range*)
Search ChemSpider by mass +/- range.

Parameters

- **mass** (*float* (https://docs.python.org/3/library/functions.html#float)) – The mass to search for.
- **mass_range** (*float* (https://docs.python.org/3/library/functions.html#float)) – The +/- mass range to allow.

Returns A list of Compounds.

Return type list (https://docs.python.org/3/library/stdtypes.html#list)[*Compound* (page 18)]

simple_search(*query*)

Search ChemSpider with arbitrary query.

A maximum of 100 results are returned. Security token is required.

Parameters **query** (*string*) – Search query - a name, SMILES, InChI, InChIKey, CSID, etc.

Returns List of Compounds.

Return type list (https://docs.python.org/3/library/stdtypes.html#list)[*Compound* (page 18)]

3.1.2 chemspipy.objects

class chemspipy.**Compound**

A class for retrieving and caching details about a specific ChemSpider record.

The purpose of this class is to provide access to various parts of the ChemSpider API that return information about a compound given its ChemSpider ID. Information is loaded lazily when requested, and cached for future access.

Parameters

- **cs** (ChemSpider (page 13)) – ChemSpider session.
- **csid** (*int* | *string*) – ChemSpider ID.

csid

ChemSpider ID.

image_url

Return the URL of a PNG image of the 2D chemical structure.

molecular_formula

Return the molecular formula for this Compound.

smiles

Return the SMILES for this Compound.

inchi

Return the InChI for this Compound.

inchikey
Return the InChIKey for this Compound.

average_mass
Return the average mass of this Compound.

molecular_weight
Return the molecular weight of this Compound.

monoisotopic_mass
Return the monoisotopic mass of this Compound.

nominal_mass
Return the nominal mass of this Compound.

alogp
Return the calculated AlogP for this Compound.

xlogp
Return the calculated XlogP for this Compound.

common_name
Return the common name for this Compound.

mol_2d
Return the MOL file for this Compound with 2D coordinates.

mol_3d
Return the MOL file for this Compound with 3D coordinates.

mol_raw
Return unprocessed MOL file for this Compound.

image
Return a 2D depiction of this Compound.

spectra
Return all the available spectral data for this Compound.

3.1.3 chemspipy.search

A wrapper for asynchronous search requests.

class chemspipy.**Results**
Container class to perform a search on a background thread and hold the results when ready.

Generally shouldn't be instantiated directly. See `search()` instead.

Parameters

- **cs** (ChemSpider (page 13)) – ChemSpider session.
- **searchfunc** (*function*) – Search function that returns a transaction ID.

- **searchargs** (*tuple* (https://docs.python.org/3/library/stdtypes.html#tuple)) – Arguments for the search function.
- **propagate** (*bool* (https://docs.python.org/3/library/functions.html#bool)) – If True, raise exceptions. If False, store on `exception` property.
- **max_requests** (*int* (https://docs.python.org/3/library/functions.html#int)) – Maximum number of times to check if search results are ready.

ready()
Return True if the search finished.

success()
Return True if the search finished with no errors.

wait()
Block until the search has completed and optionally raise any resulting exception.

status
Current status string returned by ChemSpider.

One of: 'Unknown', 'Created', 'Scheduled', 'Processing', 'Suspended', 'PartialResultReady', 'ResultReady'

exception
Any Exception raised during the search. Blocks until the search is finished.

message
A contextual message about the search. Blocks until the search is finished.

count
The number of search results. Blocks until the search is finished.

duration
The time taken to perform the search. Blocks until the search is finished.

3.1.4 chemspipy.errors

Exceptions raised by ChemSpiPy.

exception chemspipy.errors.**ChemSpiPyError**
Root ChemSpiPy Exception.

exception chemspipy.errors.**ChemSpiPyParseError**
Raised when ChemSpiPy fails to parse a response from the ChemSpider servers.

exception chemspipy.errors.**ChemSpiPyAuthError**
Raised when the security token doesn't have access to an endpoint.

exception chemspipy.errors.**ChemSpiPyNotFoundError**
Raised when no record is present for the requested CSID.

exception chemspipy.errors.**ChemSpiPyTimeoutError**
Raised when an asynchronous request times out.

exception chemspipy.errors.**ChemSpiPyServerError**
Raised when ChemSpider returns a 500 status code with an error message.

Index

M

N

R

S

W

X