ChemSpiPy Documentation

Release 1.0.5

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.

Warning: Make sure you copy the entire token from the Chemspider profile page. The text field is quite narrow so you may have to drag across to the right to select the entire token. The token format should be `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.

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-1.0.5.tar.gz
cd ChemSpiPy-1.0.5
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, `cs`.

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 26) 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 Compound

Many ChemSpiPy search methods return *Compound* (page 26) objects, which provide more functionality that a simple list of ChemSpider IDs. The primary benefit is allowing easy access to further compound properties after performing a search.

2.4.1 Creating a Compound

The easiest way to create a *Compound* (page 26) for a given ChemSpider ID is to use the `get_compound` method:

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

Alternatively, a *Compound* (page 26) can be instantiated directly:

```
>>> compound = Compound(cs, 2157)
```

Either way, no requests are made to the ChemSpider servers until specific *Compound* (page 26) properties are requested:

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

Properties are cached locally after the first time they are retrieved, speeding up subsequent access and reducing the number of unnecessary requests to the ChemSpider servers.

2.4.2 Searching for Compounds

See the *searching documentation* (page 10) for full details.

2.4.3 Compound properties

- `csid`: ChemSpider ID.
- `image_url`: URL of a PNG image of the 2D chemical structure.
- `molecular_formula`: Molecular formula.
- `smiles`: SMILES string.
- `stdinchi`: Standard InChI string.
- `stdinchikey`: Standard InChIKey.

- `inchi`: InChI string.
- `inchikey`: InChIKey.
- `average_mass`: Average mass.
- `molecular_weight`: Molecular weight.
- `monoisotopic_mass`: Monoisotopic mass.
- `nominal_mass`: Nominal mass.
- `alogp`: AlogP.
- `xlogp`: XlogP.
- `common_name`: Common Name.
- `mol_2d`: MOL file containing 2D coordinates.
- `mol_3d`: MOL file containing 3D coordinates.
- `mol_raw`: Unprocessed MOL file.
- `image`: 2D depiction as binary data in PNG format.
- `spectra`: List of spectra.

2.4.4 Implementation details

Each *Compound* (page 26) object is a simple wrapper around a ChemSpider ID. Behind the scenes, the property methods use the `get_compound_info`, `get_extended_compound_info`, `get_record_mol` and `get_compound_thumbnail` API methods to retrieve the relevant information. It is possible to use these API methods directly if required:

```
>>> info = cs.get_extended_compound_info(2157)
{u'smiles': u'CC(=O)Oc1ccccc1C(=O)O', u'common_name': u'Aspirin', u
↪'nominal_mass': 180.0, u'molecular_formula': u'C_{9}H_{8}O_{4}', u
↪'inchikey': u'BSYNRYMUTXBXSQ-UHFFFAOYAW', u'molecular_weight':␣
↪180.1574, u'inchi': u'InChI=1/C9H8O4/c1-6(10)13-8-5-3-2-4-
↪7(8)9(11)12/h2-5H,1H3,(H,11,12)', u'average_mass': 180.1574, u
↪'csid': 2157, u'alogp': 0.0, u'xlogp': 0.0, u'monoisotopic_mass':␣
↪180.042252}
```

Results are returned as a python dictionary that is derived directly from the ChemSpider API XML response.

2.5 Searching

ChemSpiPy provides a number of different ways to search ChemSpider.

2.5.1 Compound search

The main ChemSpiPy search method functions in a similar way to the main search box on the ChemSpider website. Just provide any type of query, and ChemSpider will interpret it and provide the most relevant results:

```
>>> cs.search('O=C(OCC)C')
Results([Compound(8525)])
>>> cs.search('glucose')
Results([Compound(5589), Compound(58238), Compound(71358),␣
↪Compound(96749), Compound(9312824), Compound(9484839)])
>>> cs.search('2157')
Results([Compound(2157)])
```

The supported query types include systematic names, synonyms, trade names, registry numbers, molecular formula, SMILES, InChI and InChIKey.

The *Results* (page 29) object that is returned can be treated just like any regular python list. For example, you can iterate over the results:

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

The *Results* (page 29) object also provides the time take to perform the search, and a message that explains how the query type was resolved:

```
>>> r = cs.search('Glucose')
>>> print(r.duration)
u'0:00:00.017'
>>> print(r.message)
u'Found by approved synonym'
```

2.5.2 Asynchronous searching

Certain types of search can sometimes take slightly longer, which can be inconvenient if the search method blocks the Python interpreter until the search results are returned. Fortunately, the ChemSpiPy search method works asynchronously.

Once a search is executed, ChemSpiPy immediately returns the *Results* (page 29) object, which is actually empty at first:

```
>>> results = cs.search('O=C(OCC)C')
>>> print(results.ready())
False
```

In a background thread, ChemSpiPy is making the search request and waiting for the response. But in the meantime, it is possible to continue performing other tasks in the main Python interpreter process. Call `ready()` at any point to check if the results have been returned and are available.

Any attempt to access the results will just block until the results are ready, like a simple synchronous search. To manually block the main thread until the results are ready, use the `wait()` method:

```
>>> results.wait()
>>> results.ready()
True
```

For more detailed information about the status of a search, use the `status` property:

```
>>> results.status
u'Created'
>>> results.wait()
>>> results.status
u'ResultReady'
```

The possible statuses are `Unknown`, `Created`, `Scheduled`, `Processing`, `Suspended`, `PartialResultReady`, `ResultReady`.

2.5.3 Simple search

The asynchronous search is designed to be simple as possible, but it's possible that the additional overhead might be overkill in some cases. The `simple_search` method provides a simpler synchronous alternative. Use it in the same way:

```
>>> cs.simple_search('Glucose')
[Compound(5589), Compound(58238), Compound(71358), Compound(96749),␣
↪Compound(9312824), Compound(9484839)]
```

In this case, the main Python thread will be blocked until the search results are returned, and the results actually are just in a regular Python list. A maximum of 100 results are returned.

2.5.4 Search by formula

Searching by molecular formula is supported by the main `search` method, but there is the possibility that a formula could be interpreted as a name or SMILES or another query type. To specifically search by formula, use:

```
>>> cs.search_by_formula('C44H30N4Zn')
[Compound(436642), Compound(3232330), Compound(24746832),␣
↪Compound(26995124)]
```

2.5.5 Search by mass

It is also possible to search ChemSpider by mass by specifying a certain range:

```
>>> cs.search_by_mass(680, 0.001)
[Compound(8298180), Compound(12931939), Compound(12931969),␣
↪Compound(21182158)]
```

The first parameter specifies the desired molecular mass, while the second parameter specifies the allowed ± range of values.

2.6 Spectra

Many compound records in ChemSpider have spectra associated with them.

2.6.1 Retrieving spectra

If there are spectra available for a *Compound* (page 26), you can retrieve them using the `spectra` property:

```
>>> compound = cs.get_compound(2157)
>>> print(compound.spectra)
[Spectrum(2303), Spectrum(2304), Spectrum(3558), Spectrum(6639),␣
↪Spectrum(6640), Spectrum(6641), Spectrum(6642), Spectrum(6643),␣
↪Spectrum(6644), Spectrum(6645), Spectrum(8553), Spectrum(8554)]
```

Alternatively, you can get spectra directly by using either the compound ChemSpider ID or the Spectrum ID:

```
>>> cs.get_spectrum(362)
Spectrum(362)
>>> cs.get_compound_spectra(71358)
[Spectrum(360), Spectrum(361), Spectrum(3172)]
```

2.6.2 Spectrum metadata

Each *Spectrum* (page 28) object has a number of properties:

```
>>> spectrum = cs.get_spectrum(3558)
>>> print(spectrum.spectrum_id)
3558
>>> print(spectrum.csid)
2157
>>> print(spectrum.spectrum_type)
HNMR
```

(continues on next page)

(continued from previous page)

```
>>> print(spectrum.file_name)
Spectrum_315.jdx
>>> print(spectrum.comments)
collected by David Bulger at Oral Roberts University on a JEOL 300␣
 ↪MHz NMR with methanol as the solvent
>>> print(spectrum.original_url)
http://onschallenge.wikispaces.com/Exp072
>>> print(spectrum.url)
http://www.chemspider.com/FilesHandler.ashx?type=blob&disp=1&id=3558
```

2.6.3 Spectrum data

The data file for each spectrum is also available using the data property:

```
>>> spectra = cs.get_compound_spectra(2424)
>>> caffeine_ir = spectra[8]
>>> print(caffeine_ir.data)
```

Typically this is in JCAMP-DX format.

2.7 Miscellaneous

2.7.1 Constructing API URLs

See the ChemSpider API documentation (http://www.chemspider.com/AboutServices.aspx) for more details.

```
>>> cs.construct_api_url('MassSpec', 'GetExtendedCompoundInfo',␣
 ↪csid='2157')
u'http://www.chemspider.com/MassSpec.asmx/GetExtendedCompoundInfo?
 ↪csid=2157'
```

2.7.2 Data sources

Get a list of data sources in ChemSpider:

```
>>> cs.get_databases()
['Abacipharm', 'Abblis Chemicals', 'Abcam', 'ABI Chemicals',
 ↪'Abmole Bioscience', 'ACB Blocks', 'Accela ChemBio', ... ]
```

2.8 Advanced

2.8.1 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)
```

2.8.2 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 1.0.5, Python 2.7')
```

2.8.3 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).

2.9 Contributing

Contributions of any kind are greatly appreciated!

2.9.1 Feedback

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

2.9.2 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/dev.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:

```
pytest
```

   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.

Usage:

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

Parameters

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

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 26)

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[*Compound* (page 26)]

get_spectrum(*spectrum_id*)
Return a *Spectrum* (page 28) object for a given spectrum ID. Subscriber role security token is required.

Parameters **spectrum_id** (*string|int*) – Spectrum ID.

Returns The Spectrum with the specified spectrum ID.

Return type *Spectrum* (page 28)

get_spectra(*spectrum_ids*)
Return a *Spectrum* (page 28) object for a given spectrum ID. Subscriber role security token is required.

Parameters **spectrum_ids** (*list* (https://docs.python.org/3/library/stdtypes.html#list) *[* – List of spectrum IDs.

Returns List of spectra with the specified spectrum IDs.

Return type list[*Spectrum* (page 28)]

get_compound_spectra(*csid*)
Return *Spectrum* (page 28) objects for all the spectra associated with a ChemSpider ID.

Parameters **csid** – stringlint csid: ChemSpider ID.

Returns List of spectra for the specified ChemSpider ID.

Return type list[*Spectrum* (page 28)]

get_all_spectra()
Return a full list of *Spectrum* (page 28) objects for all spectra in ChemSpider.

Subscriber role security token is required.

Returns Full list of spectra in ChemSpider.

Return type list[*Spectrum* (page 28)]

search(*query*, *order=None*, *direction=ASCENDING*, *raise_errors=False*)
Search ChemSpider for the specified query and return the results. Security token is required.

Parameters

- **query** (*string|int*) – Search query.
- **order** (*string*) – (Optional) *CSID* (page 26), *MASS_DEFECT* (page 26), *MOLECULAR_WEIGHT* (page 26), *REFERENCE_COUNT* (page 26), *DATASOURCE_COUNT* (page 26), *PUBMED_COUNT* (page 26) or *RSC_COUNT* (page 26).
- **direction** (*string*) – (Optional) *ASCENDING* (page 26) or *DESCENDING* (page 26).
- **raise_errors** (*bool* (https://docs.python.org/3/library/functions.html#bool)) – If True, raise exceptions. If False, store on Results `exception` property.

Returns Search Results list.

Return type *Results* (page 29)

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* (page 26).

Return type list[*Compound* (page 26)]

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

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

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_databases()
Get the list of datasources in ChemSpider.

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_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=MOL2D*, *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*) – *MOL2D* (page 25), *MOL3D* (page 25) or *BOTH* (page 26).
- **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_compound_spectra_info(*csid*)
Get information about all the spectra for a ChemSpider ID. Subscriber role security token is required.

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

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)]

get_spectrum_info(*spectrum_id*)
Get information for a specific spectrum ID. Subscriber role security token is required.

Parameters **spectrum_id** (*string|int*) – spectrum ID.

Returns Spectrum info.

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

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

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)]

get_all_spectra_info()
Get full list of all spectra in ChemSpider. Subscriber role security token is required.

rtype: list[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

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 (MassSpecAPI, Search, Spectra, InChI).
- **endpoint** (*string*) – ChemSpider API endpoint.
- **params** – (Optional) Parameters for the ChemSpider endpoint as keyword arguments.

Return type string

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=CSID*, *direction=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* (page 26), *MASS_DEFECT* (page 26), *MOLECULAR_WEIGHT* (page 26), *REFERENCE_COUNT* (page 26), *DATASOURCE_COUNT* (page 26), *PUBMED_COUNT* (page 26) or *RSC_COUNT* (page 26).
- **direction** (*string*) – *ASCENDING* (page 26) or *DESCENDING* (page 26).

Returns Transaction ID.

Return type string

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_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[*Compound* (page 26)]

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[*Compound* (page 26)]

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

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

Returns A list of Compounds.

Return type list[*Compound* (page 26)]

simple_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[*Compound* (page 26)]

chemspipy.api.**MOL2D = u'2d'**
2D coordinate dimensions

chemspipy.api.**MOL3D = u'3d'**
3D coordinate dimensions

chemspipy.api.**BOTH = u'both'**
Both coordinate dimensions

chemspipy.api.**ASCENDING = u'ascending'**
Ascending sort direction

chemspipy.api.**DESCENDING = u'descending'**
Descending sort direction

chemspipy.api.**CSID = u'csid'**
CSID sort order

chemspipy.api.**MASS_DEFECT = u'mass_defect'**
Mass defect sort order

chemspipy.api.**MOLECULAR_WEIGHT = u'molecular_weight'**
Molecular weight sort order

chemspipy.api.**REFERENCE_COUNT = u'reference_count'**
Reference count sort order

chemspipy.api.**DATASOURCE_COUNT = u'datasource_count'**
Datasource count sort order

chemspipy.api.**PUBMED_COUNT = u'pubmed_count'**
Pubmed count sort order

chemspipy.api.**RSC_COUNT = u'rsc_count'**
RSC count sort order

3.1.2 chemspipy.objects

Objects returned by ChemSpiPy API methods.

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 19)) – 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.

Return type string

smiles
Return the SMILES for this Compound.

Return type string

stdinchi
Return the Standard InChI for this Compound.

Return type string

stdinchikey
Return the Standard InChIKey for this Compound.

Return type string

inchi
Return the InChI for this Compound.

Return type string

inchikey
Return the InChIKey for this Compound.

Return type string

average_mass
Return the average mass of this Compound.

Return type float (https://docs.python.org/3/library/functions.html#float)

molecular_weight
Return the molecular weight of this Compound.

Return type float (https://docs.python.org/3/library/functions.html#float)

monoisotopic_mass
Return the monoisotopic mass of this Compound.

Return type float (https://docs.python.org/3/library/functions.html#float)

nominal_mass
Return the nominal mass of this Compound.

Return type float (https://docs.python.org/3/library/functions.html#float)

alogp
Return the calculated AlogP for this Compound.

Return type float (https://docs.python.org/3/library/functions.html#float)

xlogp
Return the calculated XlogP for this Compound.

Return type float (https://docs.python.org/3/library/functions.html#float)

common_name
Return the common name for this Compound.

Return type string

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

Return type string

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

Return type string

mol_raw
Return unprocessed MOL file for this Compound.

Return type string

image
Return a 2D depiction of this Compound.

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

spectra
Return all the available spectral data for this Compound.

Return type list[*Spectrum* (page 28)]

class chemspipy.**Spectrum**
A class for retrieving and caching details about a Spectrum.

Initializing a Spectrum from a spectrum ID requires a subscriber role security token.

Parameters

- **cs** (ChemSpider (page 19)) – ChemSpider session.
- **spectrum_id** (*int* / *string*) – Spectrum ID.

classmethod from_info_dict(*cs*, *info*)
Initialize a Spectrum from an info dict that has already been retrieved.

spectrum_id
Spectrum ID.

Return type int (https://docs.python.org/3/library/functions.html#int)

csid
ChemSpider ID of related compound.

Return type int (https://docs.python.org/3/library/functions.html#int)

spectrum_type
Spectrum type.

Possible values include HNMR, CNMR, IR, UV-Vis, NIR, EI, 2D1H1HCOSY, 2D1H13CD, APCI+, R, MALDI+, 2D1H13CLR, APPI-, CI+ve, ESI+, 2D1H1HOESY, FNMR, CI-ve, ESI-, PNMR.

Return type string

file_name
Spectrum file name.

Return type string

comments
Spectrum comments. Can be None.

Return type string

url
Spectrum URL.

Return type string

data
Spectrum data file contents. Requires an additional request. Result is cached.

Return type string

original_url
Original spectrum URL. Can be None.

Return type string

submitted_date
Spectrum submitted date.

Return type datetime.datetime (https://docs.python.org/3/library/datetime.html#datet

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 19)) – 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.

- **raise_errors** (*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.

Return type bool (https://docs.python.org/3/library/functions.html#bool)

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

Return type bool (https://docs.python.org/3/library/functions.html#bool)

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

status
Current status string returned by ChemSpider.

Returns 'Unknown', 'Created', 'Scheduled', 'Processing', 'Suspended', 'PartialResultReady', 'ResultReady'

Return type string

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.

Return type string

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

Return type int (https://docs.python.org/3/library/functions.html#int)

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

Return type `datetime.timedelta` (https://docs.python.org/3/library/datetime.html#dat

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

I

M

N

O

P

R

S

U

W

X