SPARCL

Release 1.2.0

S.Pothier, A.Jacques

Jun 09, 2023

CONTENTS:

This documents the Python Client for SPARCL (*SPectra Analysis and Retrievable Catalog Lab*)

Last change: Jun 09, 2023
Version: 1.2
Release: 1.2.0

CHAPTER

ONE

SPARCL PACKAGE

1.1 sparcl.client module

Client module for SPARCL. This module interfaces to the SPARC-Server to get spectra data.

class sparcl.client.**SparclClient**(*, *url='https://astrosparcl.datalab.noirlab.edu'*, *verbose=False*, *connect_timeout=1.1*, *read_timeout=5400*)

Bases: `object`

Provides interface to SPARCL Server. When using this to report a bug, set verbose to True. Also print your instance of this. The results will include important info about the Client and Server that is usefule to Developers.

Parameters

- **url** (`str`, optional) – Base URL of SPARC Server. Defaults to 'https://astrosparcl.datalab.noirlab.edu'.
- **verbose** (`bool`, optional) – Default verbosity is set to False for all client methods.
- **connect_timeout** (`float`, optional) – Number of seconds to wait to establish connection with server. Defaults to 1.1.
- **read_timeout** (`float`, optional) – Number of seconds to wait for server to send a response. Generally time to wait for first byte. Defaults to 5400.

Example

```
>>> client = SparclClient()
```

Raises

Exception – Object creation compares the version from the Server against the one expected by the Client. Throws an error if the Client is a major version or more behind.

find(*outfields=None*, *, *constraints={}*, *limit=500*, *sort=None*, *verbose=None*)

Find records in the SPARC database.

Parameters

- **outfields** (`list`, optional) – List of fields to return. Only CORE fields may be passed to this parameter. Defaults to None, which will return only the sparcl_id and _dr fields.
- **constraints** (`dict`, optional) – Key-Value pairs of constraints to place on the record selection. The Key part of the Key-Value pair is the field name and the Value part of the

Key-Value pair is a list of values. Defaults to no constraints. This will return all records in the database subject to restrictions imposed by the `limit` parameter.

- **limit** (`int`, optional) – Maximum number of records to return. Defaults to 500.
- **sort** (`list`, optional) – Comma separated list of fields to sort by. Defaults to None. (no sorting)
- **verbose** (`bool`, optional) – Set to True for in-depth return statement. Defaults to False.

Returns
Contains header and records.

Return type
Found

Example

```
>>> client = SparclClient()
>>> outs = ['id', 'ra', 'dec']
>>> cons = {'spectype': ['GALAXY'], 'redshift': [0.5, 0.9]}
>>> found = client.find(outfields=outs, constraints=cons)
>>> sorted(list(found.records[0].keys()))
['_dr', 'dec', 'id', 'ra']
```

get_all_fields(*, *dataset_list=None*)

Get fields tagged as 'all' that are in DATASET_LIST. These are the fields used for the ALL value of the include parameter of client.retrieve().

Parameters
dataset_list (`list`, optional) – List of data sets from which to get all fields. Defaults to None, which will return the intersection of all fields in all data sets hosted on the SPARC database.

Returns
List of fields tagged as 'all' from DATASET_LIST.

Example

```
>>> client = SparclClient()
>>> client.get_all_fields()
['data_release', 'datasetgroup', 'dateobs', 'dateobs_center', 'dec', 'exptime',
↪'fiberid', 'flux', 'id', 'instrument', 'ivar', 'mask', 'mjd', 'model', 'plate
↪', 'ra', 'redshift', 'redshift_err', 'redshift_warning', 'run1d', 'run2d',
↪'site', 'sky', 'specid', 'specobjid', 'specprimary', 'spectype', 'targetid',
↪'telescope', 'wave_sigma', 'wavelength', 'wavemax', 'wavemin']
```

get_available_fields(*, *dataset_list=None*)

Get subset of fields that are in all (or selected) DATASET_LIST. This may be a bigger list than will be used with the ALL keyword to client.retreive().

Parameters
dataset_list (`list`, optional) – List of data sets from which to get available fields. Defaults to None, which will return the intersection of all available fields in all data sets hosted on the SPARC database.

Returns
Set of fields available from data sets in DATASET_LIST.

Example

```
>>> client = SparclClient()
>>> sorted(client.get_available_fields())
['data_release', 'datasetgroup', 'dateobs', 'dateobs_center', 'dec', 'dirpath',
↪'exptime', 'extra_files', 'fiberid', 'filename', 'filesize', 'flux', 'id',
↪'instrument', 'ivar', 'mask', 'mjd', 'model', 'plate', 'ra', 'redshift',
↪'redshift_err', 'redshift_warning', 'run1d', 'run2d', 'site', 'sky', 'specid',
↪ 'specobjid', 'specprimary', 'spectype', 'targetid', 'telescope', 'updated',
↪'wave_sigma', 'wavelength', 'wavemax', 'wavemin']
```

get_default_fields(*, *dataset_list=None*)

Get fields tagged as 'default' that are in DATASET_LIST. These are the fields used for the DEFAULT value of the include parameter of client.retrieve().

Parameters
dataset_list (list, optional) – List of data sets from which to get the default fields. Defaults to None, which will return the intersection of default fields in all data sets hosted on the SPARC database.

Returns
List of fields tagged as 'default' from DATASET_LIST.

Example

```
>>> client = SparclClient()
>>> client.get_default_fields()
['flux', 'id', 'wavelength']
```

missing(*uuid_list*, *, *dataset_list=None*, *countOnly=False*, *verbose=False*)

Return the subset of sparcl_ids in the given uuid_list that are NOT stored in the SPARC database.

Parameters

- **uuid_list** (list) – List of sparcl_ids.
- **dataset_list** (list, optional) – List of data sets from which to find missing sparcl_ids. Defaults to None, meaning all data sets hosted on the SPARC database.
- **countOnly** (bool, optional) – Set to True to return only a count of the missing sparcl_ids from the uuid_list. Defaults to False.
- **verbose** (bool, optional) – Set to True for in-depth return statement. Defaults to False.

Returns
A list of the subset of sparcl_ids in the given uuid_list that are NOT stored in the SPARC database.

Example

```
>>> client = SparclClient()
>>> ids = ['ddbb57ee-8e90-4a0d-823b-0f5d97028076',]
>>> client.missing(ids)
['ddbb57ee-8e90-4a0d-823b-0f5d97028076']
```

missing_specids(*specid_list*, *, *dataset_list=None*, *countOnly=False*, *verbose=False*)

Return the subset of specids in the given specid_list that are NOT stored in the SPARC database.

Parameters

- **specid_list** (list) – List of specids.
- **dataset_list** (list, optional) – List of data sets from which to find missing specids. Defaults to None, meaning all data sets hosted on the SPARC database.
- **countOnly** (bool, optional) – Set to True to return only a count of the missing specids from the specid_list. Defaults to False.
- **verbose** (bool, optional) – Set to True for in-depth return statement. Defaults to False.

Returns

A list of the subset of specids in the given specid_list that are NOT stored in the SPARC database.

Example

```
>>> client = SparclClient(url=_PAT)
>>> specids = ['7972592460248666112', '3663710814482833408']
>>> client.missing_specids(specids + ['bad_id'])
['bad_id']
```

retrieve(*uuid_list*, *, *svc='spectras'*, *format='pkl'*, *include='DEFAULT'*, *dataset_list=None*, *limit=500*, *chunk=500*, *verbose=None*)

Retrieve spectra records from the SPARC database by list of sparcl_ids.

Parameters

- **uuid_list** (list) – List of sparcl_ids.
- **svc** (str, optional) – Defaults to 'spectras'.
- **format** (str, optional) – Defaults to 'pkl'.
- **include** (list, optional) – List of field names to include in each record. Defaults to 'DEFAULT', which will return the fields tagged as 'default'.
- **dataset_list** (list, optional) – List of data sets from which to retrieve spectra data. Defaults to None, meaning all data sets hosted on the SPARC database.
- **limit** (int, optional) – Maximum number of records to return. Defaults to 500.
- **chunk** (int, optional) – Size of chunks to break list into. Defaults to 500.
- **verbose** (bool, optional) – Set to True for in-depth return statement. Defaults to False.

Returns

Contains header and records.

Return type
Retrieved

Example

```
>>> client = SparclClient()
>>> ids = ['000017b6-56a2-4f87-8828-3a3409ba1083',]
>>> inc = ['id', 'flux', 'wavelength', 'model']
>>> ret = client.retrieve(uuid_list=ids, include=inc)
>>> type(ret.records[0].wavelength)
<class 'numpy.ndarray'>
```

retrieve_by_specid(*specid_list*, ***, *svc='spectras'*, *format='pkl'*, *include='DEFAULT'*, *dataset_list=None*, *limit=500*, *verbose=False*)

Retrieve spectra records from the SPARC database by list of specids.

Parameters

- **specid_list** (list) – List of specids.
- **include** (list, optional) – List of field names to include in each record. Defaults to 'DEFAULT', which will return the fields tagged as 'default'.
- **dataset_list** (list, optional) – List of data sets from which to retrieve spectra data. Defaults to None, meaning all data sets hosted on the SPARC database.
- **verbose** (bool, optional) – Set to True for in-depth return statement. Defaults to False.

Returns
Contains header and records.

Return type
Retrieved

Example

```
>>> client = SparclClient()
>>> sids = [5840097619402313728, -8985592895187431424]
>>> inc = ['specid', 'flux', 'wavelength', 'model']
>>> ret = client.retrieve_by_specid(specid_list=sids, include=inc)
>>> len(ret.records[0].wavelength)
4617
```

property version

Return version of Server Rest API used by this client. If the Rest API changes such that the Major version increases, a new version of this module will likely need to be used.

Returns
API version (float).

Example

```
>>> client = SparclClient()
>>> client.version
8.0
```

1.2 sparcl.exceptions module

exception sparcl.exceptions.**BadInclude**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

Include list contains invalid data field(s).

exception sparcl.exceptions.**BadPath**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

A field path starts with a non-core field.

exception sparcl.exceptions.**BadQuery**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

Bad find constraints.

exception sparcl.exceptions.**BadSearchConstraint**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

exception sparcl.exceptions.**BaseSparclException**(*error_message*, *error_code=None*)

Bases: Exception

Base Class for all SPARCL exceptions.

to_dict()

Convert a SPARCL exception to a python dictionary

exception sparcl.exceptions.**NoCommonIdField**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

The field name for Science id field is not common to all Data Sets

exception sparcl.exceptions.**NoIDs**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

The length of the list of original IDs passed to the reorder method was zero

exception sparcl.exceptions.**NoRecords**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

Results did not contain any records

exception sparcl.exceptions.**ReadTimeout**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

The server did not send any data in the allotted amount of time.

exception sparcl.exceptions.**ServerConnectionError**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

exception sparcl.exceptions.**TooManyRecords**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

Too many records asked for in RETRIEVE

exception sparcl.exceptions.**UnkDr**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

The Data Release is not known or not supported.

exception sparcl.exceptions.**UnknownField**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

Unknown field name for a record

exception sparcl.exceptions.**UnknownServerError**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

Client got a status response from the SPARC Server that we do not know how to decode.

exception sparcl.exceptions.**UnknownSparcl**(*error_message*, *error_code=None*)

Bases: *BaseSparclException*

Unknown SPARCL error. If this is ever raised (seen in a log) create and use a new BaseSparcException exception that is more specific.

sparcl.exceptions.**genSparclException**(*response*, *verbose=False*)

Given status from Server response.json(), which is a dict, generate a native SPARCL exception suitable for Science programs.

1.3 sparcl.Results module

Containers for results from SPARCL Server. These include results of client.retrieve() client.find().

class sparcl.Results.**Found**(*dict_list*, *client=None*)

Bases: *Results*

Holds metadata records (and header).

append(*item*)

S.append(value) – append value to the end of the sequence

clear()

Delete the contents of this collection.

property count

Number of records in this collection.

extend(*other*)

S.extend(iterable) – extend sequence by appending elements from the iterable

property ids

List of unique identifiers of matched records.

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

property info

Info about this collection. e.g. Warnings, parameters used to get the collection, etc.

insert(*i*, *item*)

S.insert(index, value) – insert value before index

pop([*index*]) → item -- remove and return item at index (default last).

Raise IndexError if list is empty or index is out of range.

property records

Records in this collection. Each record is a dictionary.

remove(*item*)

S.remove(value) – remove first occurrence of value. Raise ValueError if the value is not present.

reorder(*ids_og*)

Reorder the retrieved records to be in the same order as the original IDs passed to client.retrieve().

Parameters
ids_og (list) – List of sparcl_ids or specIDs.

Returns

Contains header and
reordered records.

none_idx (list): List of indices where record is None.

Return type
reordered (*Retrieved*)

reverse()

S.reverse() – reverse *IN PLACE*

class sparcl.Results.**Results**(*dict_list*, *client=None*)

Bases: UserList

append(*item*)

S.append(value) – append value to the end of the sequence

clear()

Delete the contents of this collection.

property count

Number of records in this collection.

extend(*other*)

S.extend(iterable) – extend sequence by appending elements from the iterable

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

property info

Info about this collection. e.g. Warnings, parameters used to get the collection, etc.

insert(*i*, *item*)

S.insert(index, value) – insert value before index

pop([*index*]) → item -- remove and return item at index (default last).

Raise IndexError if list is empty or index is out of range.

property records

Records in this collection. Each record is a dictionary.

remove(*item*)

S.remove(value) – remove first occurrence of value. Raise ValueError if the value is not present.

reorder(*ids_og*)

Reorder the retrieved records to be in the same order as the original IDs passed to client.retrieve().

Parameters
ids_og (list) – List of sparcl_ids or specIDs.

Returns

Contains header and
reordered records.

none_idx (list): List of indices where record is None.

Return type
reordered (*Retrieved*)

reverse()

S.reverse() – reverse *IN PLACE*

class sparcl.Results.**Retrieved**(*dict_list*, *client=None*)

Bases: *Results*

Holds spectra records (and header).

append(*item*)

S.append(value) – append value to the end of the sequence

clear()

Delete the contents of this collection.

property count

Number of records in this collection.

extend(*other*)

S.extend(iterable) – extend sequence by appending elements from the iterable

index(*value*[, *start*[, *stop*]]) → integer -- return first index of value.

Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

property info

Info about this collection. e.g. Warnings, parameters used to get the collection, etc.

insert(*i*, *item*)

S.insert(index, value) – insert value before index

pop([*index*]) → item -- remove and return item at index (default last).

Raise IndexError if list is empty or index is out of range.

property records

Records in this collection. Each record is a dictionary.

remove(*item*)

S.remove(value) – remove first occurrence of value. Raise ValueError if the value is not present.

reorder(*ids_og*)

Reorder the retrieved records to be in the same order as the original IDs passed to client.retrieve().

Parameters

ids_og (list) – List of sparcl_ids or specIDs.

Returns

Contains header and
reordered records.

none_idx (list): List of indices where record is None.

Return type

reordered (*Retrieved*)

reverse()

S.reverse() – reverse *IN PLACE*

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex

CHAPTER

THREE

LICENSE

PYTHON MODULE INDEX

S

INDEX

S

T

U

V