Koordinates Python API Client Documentation

Release 0.5.0

Koordinates

May 20, 2020

CONTENTS

1 Features **3**
- 1.1 Compatibility . . . 3

2 User Guide **5**
- 2.1 Introduction . . . 5
- 2.2 Installation . . . 5
- 2.3 Quick Start . . . 6
- 2.4 Authentication . . . 8
- 2.5 Pagination . . . 8
- 2.6 Limiting Results . . . 8
- 2.7 Counting Results . . . 8
- 2.8 Result Expansion . . . 9
- 2.9 Contributing . . . 9
- 2.10 Developer Interface . . . 12

3 Support **33**

Python Module Index **35**

Index **37**

Release v0.5.0.

A BSD-licensed Python client library for a number of Koordinates web APIs.

The library provides easy access to Koordinates web services, particularly the Publisher Admin APIs.:

```
import koordinates

client = koordinates.Client(host='labs.koordinates.com', token='MY_API_TOKEN')

# print the 10 most recently created layers
for layer in client.layers.order_by('created_at')[:10]:
   print(layer)
```

CHAPTER

ONE

FEATURES

The library aims to reflect the available Koordinates web APIs. Currently the following APIs have support in the library:

- Data Catalog API
- Layers & Tables API
- Sets API
- Publishing API
- Licenses API
- Metadata API
- Token API
- Data Sources API
- Permissions API
- Exports API

We're working hard to add support for additional APIs to the library, so expect this list to grow soon.

1.1 Compatibility

- Python 2.7
- Python 3.3+

CHAPTER

TWO

USER GUIDE

2.1 Introduction

2.1.1 License

The Koordinates Python Client Library is released under the terms of the BSD License.

Copyright (c) 2015, Koordinates Limited All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Koordinates nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

2.2 Installation

This part of the documentation covers the installation of the Koordinates Python Client Library.

2.2.1 Pip

Installing the Python Library is simple with pip. All you need to do is run the following in your terminal:

```
$ pip install koordinates
```

2.2.2 Getting the Code

The Koordinates Python Client Library is on GitHub.

Development occurs in the master branch, and releases are tagged and pushed to PyPI regularly.

You can either clone the public git repository:

```
$ git clone git://github.com/koordinates/python-client.git
```

Or, download the latest release.

Once you have a copy of the source, you can embed it in your Python package, or install it into your virtualenv/site-packages:

```
$ python setup.py install
```

2.2.3 Upgrading

We strongly encourage you to use the latest release of the Python Library to ensure you get the benefit of fixes, improvements, and new features. The library follows the Semantic Versioning guidelines, so all releases with the same major version number (eg. `1.x.x`) will be backwards compatible.

2.3 Quick Start

In this guide, we provide a very short overview of how the library may be used to achieve some common tasks.

Before you begin, you'll need to know the Koordinates site you're accessing (eg. labs.koordinates.com), and have a valid API token for the site, created with the scopes you need for the APIs you're using. See *Authentication* for more information. You'll also need sufficient permissions on the site to take actions (for example, creating a Layer).

First, import the Koordinates module:

```
>>> import koordinates
```

Prepare to use the library by creating a client:

```
>>> client = koordinates.Client('labs.koordinates.com', 'MY_API_TOKEN')
```

Fetch all the Layer objects via the Layers & Tables API and iterate over them:

```
>>> for layer in client.layers.list():
...     print(layer.id)
>>>
```

Fetch filtered and sorted Layer objects via the Data Catalog API and iterate over them:

```
>>> for layer in client.catalog.list().filter(license__type='cc')\
...                                   .filter(type='layer')\
...                                   .order_by('created_at'):
...     print(layer.title)
>>>
```

The results of `.list()` returns a `Query` object which is chainable. It will only make an API request once it gets iterated over or `len()` is called on it.

Fetch a single Layer object:

```
>>> # Fetch the Layer with id = 123
>>> layer = client.layers.get(123)
>>> print(layer.title)
>>>
```

Make use of the hierarchy of data within a single object exposed as Python class instances via the library:

```
>>> # Fetch the Layer with id = 123 and extract the
>>> # data.crs value
>>> layer = client.layers.get(123)
>>> print(layer.data.crs)
>>>EPSG:2193
```

Create a new Layer from existing datasources:

```
>>> layer = koordinates.Layer()
>>> layer.name = "A Test Layer"
>>> layer.group = 999
>>> layer.data = koordinates.LayerData(datasources=[123456])
>>> layer = client.layers.create(layer)
>>> print(layer.url)
```

Publish multiple objects of various types:

```
>>> # Publish a number of items, in this case one
>>> # Table and one Layer
>>> publish = koordinates.publishing.Publish()
>>> publish.add_layer_version(1111)
>>> publish.add_table_version(2222)
>>> publish.strategy = publish.STRATEGY_TOGETHER
>>> publish = client.publishing.create(publish)
>>> print(publish.url)
```

Reimport an existing Layer from its previous data sources and create a new version:

```
>>> # Take the layer with id=8888 and reimport it
>>> layer = client.layers.get(8888)
>>> layer = layer.start_update()
```

Publish a specific version of a Layer:

```
>>> # Fetch the version with id=9999 of the Layer
>>> # with id = 8888 and publish it
>>> layer_version = client.layers.get(8888).get_version(9999)
>>> layer_version.publish()
```

2.4 Authentication

To create an API token, visit the 'APIs and Web Services' page under your user menu on the site you want a token for.

Once you have an API token, you can either pass it into the *koordinates.client.Client* object when you create it, or set it in the KOORDINATES_TOKEN environment variable.

```
# Pass token explicitly
client = koordinates.Client(host='labs.koordinates.com', token='abcdef1234567890abcdef
↪')

# Token from environment variable KOORDINATES_TOKEN
client = koordinates.Client(host='labs.koordinates.com')
```

Tokens are specific to a Koordinates site. For example, a token created for `labs.koordinates.com` wouldn't be valid for another site, such as `koordinates.com`.

Tokens need to be created with scopes appropriate for the APIs you are utilising. For example, to query Sets you need a token with the `sets:read` scope, and to create or update a Set you need a token with the `sets:write` scope.

If a required scope isn't associated with the token, you will receive an `koordinates.exceptions.InvalidTokenScope` exception.

In addition to the scopes, the user or group owner of the token needs appropriate permissions for the actions they're attempting to take - for example, viewing a particular Set.

If required permissions aren't present, you will receive a *koordinates.exceptions.Forbidden* exception.

2.5 Pagination

The library handles pagination of the results of `.list()` and related methods. These methods all act as generators and transparently fetch subsequent pages of results from the APIs in the background during iteration.

2.6 Limiting Results

Limiting the results of `.list()` and related methods is available via the python slicing syntax. Only the `[:N]` slicing style is supported. For example:

```
# Limit to a maximum of three results
for layer in client.layers.list()[:3]:
    print(layer)
```

2.7 Counting Results

In order to count the results of a query or list, use `len()`. For example:

```
print(len(client.layers.list()))
print(len(client.layers.filter(license='cc')))
```

This will perform a HEAD request unless a request has already been made (via a previous call to `len()` or iteration over the results), in which case the previous cached value will be returned.

2.8 Result Expansion

To prevent additional API requests, you can get the API to expand some relations and levels of detail in responses.

Not all properties or relations can be expanded. Refer to the Koordinates API documentation for details.

Important: Using expansions may have significant performance implications for some API requests.

To expand results in a list request:

```
for object in client.catalog.list().expand():
    # object will be a detailed model instance with
    # a full set of attributes
    print(object)
```

To expand an attribute in a get request:

```
set = client.sets.get(id=123, expand='items')
# the following get_items() call will use the .expand() results
# instead of making an additional request.
print(set, len(set.get_items()))
```

2.9 Contributing

Koordinates welcomes bug reports and contributions by the community to this module. This process is intended to be as easy as possible for both contributors and the Koordinates development team.

2.9.1 Testing

The client includes a suite of unit and functional tests. These should be used to verify that your changes don't break existing functionality, and that compatibility is maintained across supported Python versions. Tests run automatically on CircleCI for branch commits and pull requests.

To run the tests you need to:

```
$ pip install -r requirements-test.txt
$ tox
```

2.9.2 Patches

All patches should be sent as a pull request on GitHub, including tests and documentation where needed. If you're fixing a bug or making a large change the patch *must* include test coverage before it will be merged.

If you're uncertain about how to write tests, take a look at some existing tests that are similar to the code you're changing.

2.9.3 Release Process

This guide describes the process to release a new version of the library. In this example, `v0.0.0`. The library follows the Semantic Versioning guidelines, so select major, minor, and patch version numbers appropriately.

Preparations

1. Close or update all tickets for the next milestone on Github..
2. Update the *minimum* required versions of dependencies in `setup.py`. Update the *exact* version of all entries in `requirements.txt`.
3. Run **`tox`** from the project root. All tests for all supported Python versions must pass:

```
$ tox
[...]
_________ summary _________
py27: commands succeeded
py34: commands succeeded
congratulations :)
```

Note: Tox will use the `requirements-test.txt` to setup the virtualenvs, so make sure you've updated it.

4. Build the Sphinx docs. Make sure there are no errors and undefined references.

```
$ make clean docs
```

Note: You will need to install dev dependancies in `requirements-dev.txt` to build documentation.

5. Check the CircleCI build is passing.
6. Update the version number in `koordinates/__init__.py` and commit:

```
$ git commit -m 'Version 0.0.0 release' koordinates/__init__.py
```

Warning: Don't tag and push the changes yet so that you can safely rollback if you need change something!

7. Create a draft release in Github with a list of changes, acknowledgements, etc.

Build and release

1. Test the release process. Build a source distribution and test it:

```
$ python setup.py sdist
$ ls dist/
koordinates-0.0.0.tar.gz
```

Try installing them:

```
$ rm -rf /tmp/koordinates-sdist  # ensure clean state
$ virtualenv /tmp/koordinates-sdist
$ /tmp/koordinates-sdist/bin/pip install dist/koordinates-0.0.0.tar.gz
$ /tmp/koordinates-sdist/bin/python
>>> import koordinates
>>> koordinates.__version__
'0.0.0'
```

2. Create or check your accounts for the *test server <https://testpypi.python.org/pypi>* and PyPI. Update your `~/.pypirc` with your credentials:

```
[distutils]
index-servers =
    pypi
    test

[test]
repository: https://test.pypi.org/legacy/
username: <test username>
password: <test password>

[pypi]
username: <production username>
password: <production password>
```

3. Upload the distributions for the new version to the test server and test the installation again:

```
$ python setup.py sdist upload -r test

$ rm -rf /tmp/koordinates-sdist  # ensure clean state
$ virtualenv /tmp/koordinates-sdist
$ /tmp/koordinates-sdist/bin/pip install -i https://testpypi.python.org/pypi --
↪extra-index-url https://pypi.python.org/pypi koordinates
```

4. Check if the package is displayed correctly: https://testpypi.python.org/pypi/koordinates
5. Upload the package to PyPI and test its installation one last time:

```
$ python setup.py sdist upload -r pypi

$ rm -rf /tmp/koordinates-sdist  # ensure clean state
$ virtualenv /tmp/koordinates-sdist
$ pip install -U koordinates
```

6. Check the package is displayed correctly: https://pypi.python.org/pypi/koordinates

Post release

1. Push your changes:

```
$ git tag -a v0.0.0 -m "Version 0.0.0"
$ git push origin v0.0.0
```

2. Activate the documentation build for the new version.
3. Make the Github release public.
4. Update related Help pages if necessary.

2.10 Developer Interface

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

2.10.1 API Client

class koordinates.client.**Client**(*host*, *token=None*, *activate_logging=False*)
Bases: object

A *Client* is used to define the host and api-version which the user wants to connect to. The user identity is also defined when *Client* is instantiated.

get_manager(*model*)
Return the active manager for the given model. :param model: Model class to look up the manager instance for. :return: Manager instance for the model associated with this client.

get_url(*datatype*, *verb*, *urltype*, *params={}*, *api_host=None*, *api_version=None*)
Returns a fully formed url

Parameters

- **datatype** – a string identifying the data the url will access.
- **verb** – the HTTP verb needed for use with the url.
- **urltype** – an adjective used to the nature of the request.
- ****params** – substitution variables for the URL.

Returns string

Return type A fully formed url.

reverse_url(*datatype*, *url*, *verb='GET'*, *urltype='single'*, *api_version=None*)
Extracts parameters from a populated URL

Parameters

- **datatype** – a string identifying the data the url accesses.
- **url** – the fully-qualified URL to extract parameters from.
- **verb** – the HTTP verb needed for use with the url.
- **urltype** – an adjective used to the nature of the request.

Returns dict

2.10.2 Catalog

Related to the Data Catalog API

class koordinates.catalog.**CatalogEntry**(***kwargs*)
Bases: koordinates.base.Model

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

class koordinates.catalog.**CatalogManager**(*client*)
Bases: koordinates.base.Manager

Accessor for querying across the site via the Catalog API.

Access via the `catalog` property of a *koordinates.client.Client* instance.

expand(**args*, ***kwargs*)
Returns an expanded Query view of the model objects. Equivalent to calling `.list().expand()`. Using expansions may have significant performance implications for some API requests. See `koordinates.base.Query.expand()`.

filter(**args*, ***kwargs*)
Returns a filtered Query view of the model objects. Equivalent to calling `.list().filter(...)`. See `koordinates.base.Query.filter()`.

get(**args*, ***kwargs*)
Fetches a Model instance determined by the value of *id*.

Parameters **id** – numeric ID for the Model.

list()
The published version of each layer, table, set, document or source. If something hasn't been published yet, it won't appear here.

list_latest()
A filterable list view of layers, tables, sets, documents and sources, similar to *koordinates.catalog.CatalogManager.list()*. This returns the latest version of each item, regardless of whether or not it has been published.

model
alias of *CatalogEntry*

order_by(**args*, ***kwargs*)
Returns an ordered Query view of the model objects. Equivalent to calling `.list().order_by(...)`. See `koordinates.base.Query.order_by()`.

2.10.3 Layers & Tables

Related to the Layers & Tables API

class koordinates.layers.**Layer**(***kwargs*)
Bases: koordinates.base.Model, koordinates.permissions.PermissionObjectMixin

Represents a version of a single Layer or Table.

create_draft_version()
Creates a new draft version from this model content.

If anything in the data object has changed then an import will begin immediately. Otherwise to force a re-import from the previous sources call *koordinates.layers.Layer.start_import()*.

Return type *Layer*

Returns the new version

Raises ***Conflict*** – if there is already a draft version for this layer.

delete_layer()
Delete this layer.

delete_version(*version_id=None*)
Deletes this draft version (revert to published)

Raises

- ***NotAllowed*** – if this version is already published.
- ***Conflict*** – if this version is already deleted.

get_draft_version(*expand=[]*)
Get the current draft version of this layer. :raises NotFound: if there is no draft version.

get_published_version(*expand=[]*)
Get the latest published version of this layer. :raises NotFound: if there is no published version.

get_version(*version_id*, *expand=[]*)
Get a specific version of this layer

property is_draft_version
Return if this version is the draft version of a layer

property is_published_version
Return if this version is the published version of a layer

list_versions()
Filterable list of versions of a layer, always ordered newest to oldest.

If the version's source supports revisions, you can get a specific revision using `.filter(data__source__revision=value)`. Specific values depend on the source type. Use `data__source_revision__lt` or `data__source_revision__gte` to filter using < or >= operators respectively.

publish(*version_id=None*)
Creates a publish task just for this version, which publishes as soon as any import is complete.

Returns the publish task

Return type *Publish*

Raises ***Conflict*** – If the version is already published, or already has a publish job.

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

save(*with_data=False*)
Edits this draft layerversion. # If anything in the data object has changed, cancel any existing import and start a new one.

Parameters **with_data** (*bool*) – if `True`, send the data object, which will start a new import and cancel any existing one. If `False`, the data object will *not* be sent, and no import will start.

Raises ***NotAllowed*** – if the version is already published.

set_metadata(*fp*, *version_id=None*)
Set the XML metadata on this draft version.

Parameters **fp** (*file*) – file-like object to read the XML metadata from.

Raises ***NotAllowed*** – if this version is already published.

start_import(*version_id=None*)
Starts importing this draft layerversion (cancelling any running import), even if the data object hasn't changed from the previous version.

Raises ***Conflict*** – if this version is already published.

start_update()
A shortcut to create a new version and start importing it. Effectively the same as *create_draft_version()* followed by *koordinates.layers.Layer.start_import()*.

Return type *Layer*

Returns the new version

Raises ***Conflict*** – if there is already a draft version for this layer.

class koordinates.layers.**LayerManager**(*client*)
Bases: koordinates.base.Manager

Accessor for querying Layers & Tables.

Access via the `layers` property of a *koordinates.client.Client* instance.

create(*layer*)
Creates a new layer. All attributes except `name` and `data.datasources` are optional. :return: the new draft version of the layer.

create_draft(*layer_id*)
Creates a new draft version.

If anything in the data object has changed then an import will begin immediately. Otherwise to force a re-import from the previous sources call *koordinates.layers.LayerManager.start_import()*.

Return type *Layer*

Returns the new version

Raises ***Conflict*** – if there is already a draft version for this layer.

expand(**args*, ***kwargs*)
Returns an expanded Query view of the model objects. Equivalent to calling `.list().expand()`. Using expansions may have significant performance implications for some API requests. See `koordinates.base.Query.expand()`.

filter(**args*, ***kwargs*)
Returns a filtered Query view of the model objects. Equivalent to calling `.list().filter(...)`. See `koordinates.base.Query.filter()`.

get(*id*, *expand=[]*)
Fetches a Model instance determined by the value of *id*.

Parameters **id** – numeric ID for the Model.

get_draft(*layer_id*, *expand=[]*)
Get the current draft version of a layer. :raises NotFound: if there is no draft version.

get_published(*layer_id*, *expand=[]*)
Get the latest published version of this layer. :raises NotFound: if there is no published version.

get_version(*layer_id*, *version_id*, *expand=[]*)
Get a specific version of a layer.

list()
Fetches a set of model objects

Return type koordinates.base.Query

list_drafts()
A filterable list views of layers, returning the draft version of each layer. If the most recent version of a layer or table has been published already, it won't be returned here.

list_versions(*layer_id*)
Filterable list of versions of a layer, always ordered newest to oldest.

If the version's source supports revisions, you can get a specific revision using .filter(data__source__revision=value). Specific values depend on the source type. Use data__source_revision__lt or data__source_revision__gte to filter using < or >= operators respectively.

model
alias of *Layer*

order_by(**args*, ***kwargs*)
Returns an ordered Query view of the model objects. Equivalent to calling .list().order_by(...). See koordinates.base.Query.order_by().

set_metadata(*layer_id*, *version_id*, *fp*)
Set the XML metadata on a layer draft version.

Parameters **fp** (*file*) – file-like object to read the XML metadata from.

Raises ***NotAllowed*** – if the version is already published.

start_import(*layer_id*, *version_id*)
Starts importing the specified draft version (cancelling any running import), even if the data object hasn't changed from the previous version.

start_update(*layer_id*)
A shortcut to create a new version and start importing it. Effectively the same as *koordinates.layers.LayerManager.create_draft()* followed by *koordinates.layers.LayerManager.start_import()*.

class koordinates.layers.**LayerVersion**(***kwargs*)
Bases: koordinates.base.InnerModel

Represents the version property of a *Layer* instance.

class koordinates.layers.**LayerVersionManager**(*client*, *parent_manager*)
Bases: koordinates.base.InnerManager

model
alias of *LayerVersion*

class koordinates.layers.**LayerData**(***kwargs*)
Bases: koordinates.base.InnerModel

Represents the data property of a *Layer* instance.

class koordinates.layers.**LayerDataManager**(*client*, *parent_manager*)
Bases: koordinates.base.InnerManager

model
alias of *LayerData*

2.10.4 Licenses

Related to the License API

class koordinates.licenses.**LicenseManager**(*client*)
Bases: koordinates.base.Manager

Accessor for querying licenses.

Access via the `licenses` property of a *koordinates.client.Client* instance.

expand(**args*, ***kwargs*)
Returns an expanded Query view of the model objects. Equivalent to calling `.list().expand()`. Using expansions may have significant performance implications for some API requests. See `koordinates.base.Query.expand()`.

filter(**args*, ***kwargs*)
Returns a filtered Query view of the model objects. Equivalent to calling `.list().filter(...)`. See `koordinates.base.Query.filter()`.

get(*id*, *expand=[]*)
Fetches a Model instance determined by the value of *id*.

Parameters **id** – numeric ID for the Model.

get_creative_commons(*slug*, *jurisdiction=None*)
Returns the Creative Commons license for the given attributes.

Parameters

- **slug** (*str*) – the type of Creative Commons license. It must start with `cc-by` and can optionally contain `nc` (non-commercial), `sa` (share-alike), `nd` (no derivatives) terms, seperated by hyphens. Note that a CC license cannot be both `sa` and `nd`
- **jurisdiction** (*str*) – The jurisdiction for a ported Creative Commons license (eg. `nz`), or `None` for unported/international licenses.

Return type *License*

list()
Fetches a set of model objects

Return type `koordinates.base.Query`

model
alias of *License*

order_by(**args*, ***kwargs*)
Returns an ordered Query view of the model objects. Equivalent to calling `.list().order_by(...)`. See `koordinates.base.Query.order_by()`.

class koordinates.licenses.**License**(***kwargs*)
Bases: koordinates.base.Model

Represents a license that can be applied to layers, tables, sources, and documents.

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

2.10.5 Metadata

Related to the Metadata API

class koordinates.metadata.**MetadataManager**(*client*, *parent_manager*)
Bases: koordinates.base.InnerManager

Accessor for querying and updating metadata.

Access via the `metadata` property of *koordinates.layers.Layer* or *koordinates.sets.Set* instances.

model
alias of *Metadata*

set(*parent_url*, *fp*)
If the parent object already has XML metadata, it will be overwritten.

Accepts XML metadata in any of the three supported formats. The format will be detected from the XML content.

The Metadata object becomes invalid after setting

Parameters **fp** (*file*) – A reference to an open file-like object which the content will be read from.

class koordinates.metadata.**Metadata**(***kwargs*)
Bases: koordinates.base.InnerModel

get_formats()
Return the available format names for this metadata

get_xml(*fp*, *format='native'*)
Returns the XML metadata for this source, converted to the requested format. Converted metadata may not contain all the same information as the native format.

Parameters

- **fp** (*file*) – A path, or an open file-like object which the content should be written to.
- **format** (*str*) – desired format for the output. This should be one of the available formats from *get_formats()*, or `FORMAT_NATIVE` for the native format.

If you pass this function an open file-like object as the fp parameter, the function will not close that file for you.

2.10.6 Publishing

Related to the Publishing API

class koordinates.publishing.**PublishManager**(*client*)
Bases: koordinates.base.Manager

Accessor for querying Publish groups.

Access via the `publishing` property of a *koordinates.client.Client* instance.

create(*publish*)
Creates a new publish group.

expand(**args*, ***kwargs*)
Returns an expanded Query view of the model objects. Equivalent to calling `.list().expand()`.

Using expansions may have significant performance implications for some API requests. See `koordinates.base.Query.expand()`.

filter(**args*, ***kwargs*)
Returns a filtered Query view of the model objects. Equivalent to calling `.list().filter(...)`. See `koordinates.base.Query.filter()`.

get(*id*, *expand=[]*)
Fetches a Model instance determined by the value of *id*.

Parameters **id** – numeric ID for the Model.

list()
Fetches a set of model objects

Return type `koordinates.base.Query`

model
alias of *Publish*

order_by(**args*, ***kwargs*)
Returns an ordered Query view of the model objects. Equivalent to calling `.list().order_by(...)`. See `koordinates.base.Query.order_by()`.

class koordinates.publishing.**Publish**(***kwargs*)
Bases: `koordinates.base.Model`

Represents an active publishing group.

add_layer_item(*layer*)
Adds a Layer to the publish group.

add_table_item(*table*)
Adds a Table to the publish group.

cancel()
Cancel a pending publish task

get_items()
Return the item models associated with this Publish group.

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

2.10.7 Sets

Related to the Sets API

class koordinates.sets.**SetManager**(*client*)
Bases: `koordinates.base.Manager`

Accessor for querying Sets.

Access via the `sets` property of a *koordinates.client.Client* instance.

create(*set*)
Creates a new Set.

create_draft(*set_id*)
Creates a new draft version.

Return type *Client*

Returns the new version

Raises **409 Conflict** – if there is already a draft version for this set.

expand(**args*, ***kwargs*)
Returns an expanded Query view of the model objects. Equivalent to calling `.list().expand()`. Using expansions may have significant performance implications for some API requests. See `koordinates.base.Query.expand()`.

filter(**args*, ***kwargs*)
Returns a filtered Query view of the model objects. Equivalent to calling `.list().filter(...)`. See `koordinates.base.Query.filter()`.

get(*id*, *expand=[]*)
Fetches a Model instance determined by the value of *id*.

Parameters **id** – numeric ID for the Model.

get_draft(*set_id*, *expand=[]*)
Get the current draft version of a set. :raises NotFound: if there is no draft version.

get_published(*set_id*, *expand=[]*)
Get the latest published version of this set. :raises NotFound: if there is no published version.

get_version(*set_id*, *version_id*, *expand=[]*)
Get a specific version of a set.

list()
Fetches a set of model objects

Return type `koordinates.base.Query`

list_drafts()
A filterable list view of sets, returning the draft version of each set. If the most recent version has been published already, it won't be returned here.

list_versions(*set_id*)
Filterable list of versions of a set, always ordered newest to oldest.

If the version's source supports revisions, you can get a specific revision using `.filter(data__source_revision=value)`. Specific values depend on the source type. Use `data__source_revision__lt` or `data__source_revision__gte` to filter using < or >= operators respectively.

model
alias of *Set*

order_by(**args*, ***kwargs*)
Returns an ordered Query view of the model objects. Equivalent to calling `.list().order_by(...)`. See `koordinates.base.Query.order_by()`.

set_metadata(*set_id*, *fp*)
Set the XML metadata on a set.

Parameters **fp** (*file*) – file-like object to read the XML metadata from.

class koordinates.sets.**Set**(***kwargs*)
Bases: `koordinates.base.Model`, `koordinates.permissions.PermissionObjectMixin`

Represents a single set grouping of layers, tables, and documents.

delete_version(*version_id=None*)
Deletes this draft version (revert to published)

Raises

- ***NotAllowed*** – if this version is already published.
- ***Conflict*** – if this version is already deleted.

get_draft_version(*expand=()*)
Get the current draft version of this set. :raises NotFound: if there is no draft version.

get_published_version(*expand=()*)
Get the latest published version of this set. :raises NotFound: if there is no published version.

get_version(*version_id*, *expand=()*)
Get a specific version of this set

property is_draft_version
Return if this version is the draft version of a layer

property is_published_version
Return if this version is the published version of a layer

list_versions()
Filterable list of versions of a set, always ordered newest to oldest.

If the version's source supports revisions, you can get a specific revision using `.filter(data__source_revision=value)`. Specific values depend on the source type. Use `data__source_revision__lt` or `data__source_revision__gte` to filter using < or >= operators respectively.

publish(*version_id=None*)
Creates a publish task for this version.

Returns the publish task

Return type *Publish*

Raises ***Conflict*** – If the version is already published, or already has a publish job.

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

save()
Edits this draft version.

Raises ***NotAllowed*** – if the version is already published.

set_metadata(*fp*, *version_id=None*)
Set the XML metadata on this draft version.

Parameters **fp** (*file*) – file-like object to read the XML metadata from.

Raises ***NotAllowed*** – if this version is already published.

class koordinates.sets.**SetVersion**(***kwargs*)
Bases: koordinates.base.InnerModel

Represents the `version` property of a *koordinates.client.Client* instance.

class koordinates.sets.**SetVersionManager**(*client*, *parent_manager*)
Bases: koordinates.base.InnerManager

model
alias of *SetVersion*

class koordinates.sets.**SetDataManager**(*client*, *parent_manager*)
Bases: koordinates.base.InnerManager

model
alias of *SetData*

class koordinates.sets.**SetData**(***kwargs*)
Bases: koordinates.base.InnerModel

Represents the `data` property of a *koordinates.client.Client* instance.

2.10.8 Users & Groups

class koordinates.users.**UserManager**(*client*)
Bases: koordinates.base.Manager

expand(**args*, ***kwargs*)
Returns an expanded Query view of the model objects. Equivalent to calling `.list().expand()`. Using expansions may have significant performance implications for some API requests. See `koordinates.base.Query.expand()`.

filter(**args*, ***kwargs*)
Returns a filtered Query view of the model objects. Equivalent to calling `.list().filter(...)`. See `koordinates.base.Query.filter()`.

get(*id*, *expand=[]*)
Fetches a Model instance determined by the value of *id*.

Parameters **id** – numeric ID for the Model.

list()
Fetches a set of model objects

Return type `koordinates.base.Query`

model
alias of *User*

order_by(**args*, ***kwargs*)
Returns an ordered Query view of the model objects. Equivalent to calling `.list().order_by(...)`. See `koordinates.base.Query.order_by()`.

class koordinates.users.**User**(***kwargs*)
Bases: koordinates.base.Model

Represents a Koordinates User

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

class koordinates.users.**GroupManager**(*client*)
Bases: koordinates.base.Manager

expand(**args*, ***kwargs*)
Returns an expanded Query view of the model objects. Equivalent to calling `.list().expand()`. Using expansions may have significant performance implications for some API requests. See `koordinates.base.Query.expand()`.

filter(**args*, ***kwargs*)
Returns a filtered Query view of the model objects. Equivalent to calling `.list().filter(...)`. See `koordinates.base.Query.filter()`.

get(*id*, *expand=[]*)
Fetches a Model instance determined by the value of *id*.

Parameters **id** – numeric ID for the Model.

list()
Fetches a set of model objects

Return type koordinates.base.Query

model
alias of *Group*

order_by(**args*, ***kwargs*)
Returns an ordered Query view of the model objects. Equivalent to calling `.list().order_by(...)`. See `koordinates.base.Query.order_by()`.

class koordinates.users.**Group**(***kwargs*)
Bases: koordinates.base.Model

Represents a Koordinates Group

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

2.10.9 Data Sources

Related to the Data Sources API

class koordinates.sources.**SourceManager**(*client*)
Accessor for querying Sources.

Access via the `sources` property of a *koordinates.client.Client* instance.

create(*source*, *upload_progress_callback=None*)
Creates a new source.

Parameters

- **source** (*str*) – The populated Source object to create.
- **upload_progress_callback** (*function*) – For an UploadSource object, an optional callback function which receives upload progress notifications. The function should take two arguments: the number of bytes sent, and the total number of bytes to send.

Return type *Source*

expand(**args*, ***kwargs*)
Returns an expanded Query view of the model objects. Equivalent to calling `.list().expand()`. Using expansions may have significant performance implications for some API requests. See `koordinates.base.Query.expand()`.

filter(**args*, ***kwargs*)
Returns a filtered Query view of the model objects. Equivalent to calling `.list().filter(...)`. See `koordinates.base.Query.filter()`.

get(*id*, *expand=[]*)
Fetches a Model instance determined by the value of *id*.

Parameters **id** – numeric ID for the Model.

get_datasource(*source_id*, *datasource_id*)
Get a Datasource object

Return type *Datasource*

get_scan(*source_id*, *scan_id*)
Get a Scan object

Return type *Scan*

get_scan_log_lines(*source_id*, *scan_id*)
Get the log text for a Scan

Return type Iterator over log lines.

list()
Fetches a set of model objects

Return type `koordinates.base.Query`

list_datasources(*source_id*)
Filterable list of Datasources for a Source.

list_scans(*source_id=None*)
Filterable list of Scans for a Source. Ordered newest to oldest by default

model
alias of *Source*

order_by(**args*, ***kwargs*)
Returns an ordered Query view of the model objects. Equivalent to calling `.list().order_by(...)`. See `koordinates.base.Query.order_by()`.

start_scan(*source_id*)
Start a new scan of a Source.

Return type *Scan*

class `koordinates.sources.`**Source**(***kwargs*)
A source points to a place where Koordinates can get data from. Sources can contain any number of datasources.

delete()
Delete this source

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

save(*with_data=False*)
Edits this Source

class koordinates.sources.**UploadSource**(**args*, ***kwargs*)
Bases: *koordinates.sources.Source*

Subclass of Source used for uploads of files and archives, which are automatically scanned.

Example

```
>>> upload = koordinates.UploadSource()
>>> upload.user = 5
>>> upload.title = "upload_source example"
>>> upload.add_file('/path/to/data.zip')
>>> upload = client.sources.create(upload)
```

add_file(*fp*, *upload_path=None*, *content_type=None*)
Add a single file or archive to upload.

To add metadata records with a file, add a .xml file with the same upload path basename eg. `points-with-metadata.geojson` & `points-with-metadata.xml` Datasource XML must be in one of these three formats:

- ISO 19115/19139
- FGDC CSDGM
- Dublin Core (OAI-PMH)

Parameters

- **fp** (*str or file*) – File to upload into this source, can be a path or a file-like object.
- **upload_path** (*str*) – relative path to store the file as within the source (eg. `folder/0001.tif`). By default it will use `fp`, either the filename from a path or the `.name` attribute of a file-like object.
- **content_type** (*str*) – Content-Type of the file. By default it will attempt to auto-detect from the file/upload_path.

delete()
Delete this source

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

save(*with_data=False*)
Edits this Source

class koordinates.sources.**Scan**(***kwargs*)
A scan operation examines a source to find out about what datasources the source provides.

cancel()
Cancel a running Scan.

get_log_lines()
Get the log text for a scan object

Return type Iterator over log lines.

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

class koordinates.sources.**Datasource**(***kwargs*)
A datasource is a single dataset from a source. One or more datasources may be imported to create a layer, table or document.

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

2.10.10 Exports

Related to the Exports API

class koordinates.exports.**ExportManager**(*client*)
Accessor for querying and creating Exports.

Access via the `exports` property of a *koordinates.client.Client* instance.

create(*export*)
Create and start processing a new Export.

Parameters **export** (Export) – The Export to create.

Return type *Export*

property croplayers
Returns a manager for querying and listing CropLayer models

Return type *CropLayerManager*

expand(**args*, ***kwargs*)
Returns an expanded Query view of the model objects. Equivalent to calling `.list().expand()`. Using expansions may have significant performance implications for some API requests. See `koordinates.base.Query.expand()`.

filter(**args*, ***kwargs*)
Returns a filtered Query view of the model objects. Equivalent to calling `.list().filter(...)`. See `koordinates.base.Query.filter()`.

get(*id*, *expand=[]*)
Fetches a Model instance determined by the value of *id*.

Parameters **id** – numeric ID for the Model.

get_formats()
Returns a dictionary of format options keyed by data kind.

```
{
    "vector": {
        "application/x-ogc-gpkg": "GeoPackage",
        "application/x-zipped-shp": "Shapefile",
        #...
    },
    "table": {
        "text/csv": "CSV (text/csv)",
        "application/x-ogc-gpkg": "GeoPackage",
        #...
    },
    "raster": {
        "image/jpeg": "JPEG",
        "image/jp2": "JPEG2000",
        #...
    },
    "grid": {
        "application/x-ogc-aaigrid": "ASCII Grid",
        "image/tiff;subtype=geotiff": "GeoTIFF",
        #...
    },
    "rat": {
        "application/x-erdas-hfa": "ERDAS Imagine",
        #...
    }
}
```

Return type dict

list()
Fetches a set of model objects

Return type koordinates.base.Query

model
alias of *Export*

order_by(**args*, ***kwargs*)
Returns an ordered Query view of the model objects. Equivalent to calling .list().order_by(...). See koordinates.base.Query.order_by().

validate(*export*)
Validates an Export.

Parameters **export** (Export) –

Return type *ExportValidationResponse*

class koordinates.exports.**Export**(***kwargs*)
An export is a request to extract data from a Koordinates site into an archive for downloading

Example

```
>>> export = koordinates.Export()
>>> export.crs = "EPSG:4326"
>>> export.formats = {
        "vector": "application/x-zipped-shp"
    }
```

(continues on next page)

(continued from previous page)

```
>>> export.add_item(layer)
>>> client.exports.create(export)
```

add_item(*item*, ***options*)
Add a layer or table item to the export.

Parameters **item** (*Layer | Table*) – The Layer or Table to add

Return type self

cancel()
Cancel the export processing

download(*path*, *progress_callback=None*, *chunk_size=1048576*)
Download the export archive.

> **Warning:** If you pass this function an open file-like object as the `path` parameter, the function will not close that file for you.

If a `path` parameter is a directory, this function will use the Export name to determine the name of the file (returned). If the calculated download file path already exists, this function will raise a DownloadError.

You can also specify the filename as a string. This will be passed to the built-in `open()` and we will read the content into the file.

Instead, if you want to manage the file object yourself, you need to provide either a `io.BytesIO` object or a file opened with the *'b'* flag. See the two examples below for more details.

Parameters

- **path** – Either a string with the path to the location to save the response content, or a file-like object expecting bytes.
- **progress_callback** (*function*) – An optional callback function which receives upload progress notifications. The function should take two arguments: the number of bytes recieved, and the total number of bytes to recieve.
- **chunk_size** (*int*) – Chunk size in bytes for streaming large downloads and progress reporting. 1MB by default

:returns The name of the automatic filename that would be used. :rtype: str

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

class koordinates.exports.**ExportValidationResponse**(***kwargs*)
Repsonse returned by Export validation requests.

class koordinates.exports.**CropLayerManager**(*client*)
Bases: koordinates.base.Manager

Accessor for querying Crop Layers.

Access via the `exports.croplayers` property of a *koordinates.client.Client* instance.

expand(**args*, ***kwargs*)
Returns an expanded Query view of the model objects. Equivalent to calling `.list().expand()`. Using expansions may have significant performance implications for some API requests. See `koordinates.base.Query.expand()`.

filter(**args*, ***kwargs*)
Returns a filtered Query view of the model objects. Equivalent to calling `.list().filter(...)`. See `koordinates.base.Query.filter()`.

get(*id*, *expand=[]*)
Fetches a Model instance determined by the value of *id*.

Parameters **id** – numeric ID for the Model.

get_feature(*croplayer_id*, *cropfeature_id*)
Gets a crop feature

Parameters

- **croplayer_id** (*int*) – ID of a cropping layer
- **cropfeature_id** (*int*) – ID of a cropping feature

Return type *CropFeature*

list()
Fetches a set of model objects

Return type `koordinates.base.Query`

model
alias of *CropLayer*

order_by(**args*, ***kwargs*)
Returns an ordered Query view of the model objects. Equivalent to calling `.list().order_by(...)`. See `koordinates.base.Query.order_by()`.

class koordinates.exports.**CropLayer**(***kwargs*)
A crop layer provides features that can be used to crop exports to a geographic extent.

get_feature(*cropfeature_id*)
Gets a crop feature

Parameters **cropfeature_id** (*int*) – ID of a cropping feature

Return type *CropFeature*

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

class koordinates.exports.**CropFeatureManager**(*client*)
Accessor for querying Crop Features.

expand(**args*, ***kwargs*)
Returns an expanded Query view of the model objects. Equivalent to calling `.list().expand()`. Using expansions may have significant performance implications for some API requests. See `koordinates.base.Query.expand()`.

filter(**args*, ***kwargs*)
Returns a filtered Query view of the model objects. Equivalent to calling `.list().filter(...)`. See `koordinates.base.Query.filter()`.

get(*id*, *expand=[]*)
Fetches a Model instance determined by the value of *id*.

Parameters **id** – numeric ID for the Model.

list()
Fetches a set of model objects

Return type `koordinates.base.Query`

model
alias of *CropFeature*

order_by(**args*, ***kwargs*)
Returns an ordered Query view of the model objects. Equivalent to calling `.list().order_by(...)`. See `koordinates.base.Query.order_by()`.

class koordinates.exports.**CropFeature**(***kwargs*)
A crop feature provides complex pre-defined geographic extents for cropping and clipping Exports.

refresh()
Refresh this model from the server.

Updates attributes with the server-defined values. This is useful where the Model instance came from a partial response (eg. a list query) and additional details are required.

Existing attribute values will be overwritten.

2.10.11 Exception Classes

exception koordinates.exceptions.**KoordinatesException**(*message*, ***kwargs*)
Base class for all koordinates module errors

exception koordinates.exceptions.**ClientError**(*message*, ***kwargs*)
Base class for client errors

exception koordinates.exceptions.**ClientValidationError**(*message*, ***kwargs*)
Client-side validation error

exception koordinates.exceptions.**InvalidAPIVersion**(*message*, ***kwargs*)
Invalid API Version

exception koordinates.exceptions.**ServerError**(*message=None*, *error=None*, *response=None*)
Base class for errors returned from the API Servers

exception koordinates.exceptions.**BadRequest**(*message=None*, *error=None*, *response=None*)
Invalid request data or parameters. Check your request. (400)

exception koordinates.exceptions.**AuthenticationError**(*message=None*, *error=None*, *response=None*)
The API token is invalid or expired. (401)

exception koordinates.exceptions.**Forbidden**(*message=None*, *error=None*, *response=None*)
The API token or user doesn't have access to perform the request. (403)

exception koordinates.exceptions.**NotFound**(*message=None*, *error=None*, *response=None*)
The requested object was not found. (404)

exception koordinates.exceptions.**NotAllowed**(*message=None*, *error=None*, *response=None*)
The requested action isn't available for this object. (405)

exception koordinates.exceptions.**Conflict**(*message=None*, *error=None*, *response=None*)
The requested action isn't available for this object due to a conflict. (409)

exception koordinates.exceptions.**RateLimitExceeded**(*message=None*, *error=None*, *response=None*)
The request has exceeded the API rate limit. Retry the request again later. (429)

exception koordinates.exceptions.**InternalServerError**(*message=None*, *error=None*, *response=None*)
An internal server error has occurred. (500)

exception koordinates.exceptions.**ServiceUnvailable**(*message=None*, *error=None*, *response=None*)
The Koordinates service is currently unavailable. (502/503/504)

2.10.12 Permissions

Related to the Permissions API

class koordinates.permissions.**PermissionManager**(*client*, *parent_object*)
Bases: koordinates.base.InnerManager

Accessor for querying and updating permissions.

Access via the `permissions` property of *koordinates.layers.Layer* or *koordinates.sets.Set* instances.

create(*permission*)
Create single permission for the given object.

Parameters **permission** (Permission) – A single Permission object to be set.

get(*permission_id*, *expand=[]*)
List a specific permisison for the given object.

Parameters **permission_id** (*str*) – the id of the Permission to be listed.

list()
List permissions for the given object.

model
alias of *Permission*

set(*permissions*)
Set the object permissions. If the parent object already has permissions, they will be overwritten.

Parameters **permissions** (*[]*) – A group of Permission objects to be set.

class koordinates.permissions.**Permission**(***kwargs*)
Bases: koordinates.base.InnerModel

Represents a permissions for a specific *koordinates.layers.Layer* or *koordinates.sets.Set* instance.

CHAPTER

THREE

SUPPORT

Please report bugs as Github issues, or see *Contributing* if you wish to suggest an improvement or make a change. For general technical support for the APIs and library, please contact us via help.koordinates.com.

PYTHON MODULE INDEX

k

koordinates, 22
koordinates.exceptions, 30

INDEX

A

add_file() (*koordinates.sources.UploadSource method*), 25
add_item() (*koordinates.exports.Export method*), 28
add_layer_item() (*koordinates.publishing.Publish method*), 19
add_table_item() (*koordinates.publishing.Publish method*), 19
AuthenticationError, 30

B

BadRequest, 30

C

cancel() (*koordinates.exports.Export method*), 28
cancel() (*koordinates.publishing.Publish method*), 19
cancel() (*koordinates.sources.Scan method*), 25
CatalogEntry (*class in koordinates.catalog*), 12
CatalogManager (*class in koordinates.catalog*), 12
Client (*class in koordinates.client*), 12
ClientError, 30
ClientValidationError, 30
Conflict, 31
create() (*koordinates.exports.ExportManager method*), 26
create() (*koordinates.layers.LayerManager method*), 15
create() (*koordinates.permissions.PermissionManager method*), 31
create() (*koordinates.publishing.PublishManager method*), 18
create() (*koordinates.sets.SetManager method*), 19
create() (*koordinates.sources.SourceManager method*), 23
create_draft() (*koordinates.layers.LayerManager method*), 15
create_draft() (*koordinates.sets.SetManager method*), 19
create_draft_version() (*koordinates.layers.Layer method*), 13
CropFeature (*class in koordinates.exports*), 30
CropFeatureManager (*class in koordinates.exports*), 29
CropLayer (*class in koordinates.exports*), 29
CropLayerManager (*class in koordinates.exports*), 28
croplayers() (*koordinates.exports.ExportManager property*), 26

D

Datasource (*class in koordinates.sources*), 26
delete() (*koordinates.sources.Source method*), 24
delete() (*koordinates.sources.UploadSource method*), 25
delete_layer() (*koordinates.layers.Layer method*), 13
delete_version() (*koordinates.layers.Layer method*), 13
delete_version() (*koordinates.sets.Set method*), 21
download() (*koordinates.exports.Export method*), 28

E

expand() (*koordinates.catalog.CatalogManager method*), 13
expand() (*koordinates.exports.CropFeatureManager method*), 29
expand() (*koordinates.exports.CropLayerManager method*), 28
expand() (*koordinates.exports.ExportManager method*), 26
expand() (*koordinates.layers.LayerManager method*), 15
expand() (*koordinates.licenses.LicenseManager method*), 17
expand() (*koordinates.publishing.PublishManager method*), 18
expand() (*koordinates.sets.SetManager method*), 20
expand() (*koordinates.sources.SourceManager method*), 23
expand() (*koordinates.users.GroupManager method*), 23
expand() (*koordinates.users.UserManager method*), 22
Export (*class in koordinates.exports*), 27

ExportManager (*class in koordinates.exports*), 26
ExportValidationResponse (*class in koordinates.exports*), 28

F

filter() (*koordinates.catalog.CatalogManager method*), 13
filter() (*koordinates.exports.CropFeatureManager method*), 29
filter() (*koordinates.exports.CropLayerManager method*), 29
filter() (*koordinates.exports.ExportManager method*), 26
filter() (*koordinates.layers.LayerManager method*), 15
filter() (*koordinates.licenses.LicenseManager method*), 17
filter() (*koordinates.publishing.PublishManager method*), 19
filter() (*koordinates.sets.SetManager method*), 20
filter() (*koordinates.sources.SourceManager method*), 24
filter() (*koordinates.users.GroupManager method*), 23
filter() (*koordinates.users.UserManager method*), 22
Forbidden, 30

G

get() (*koordinates.catalog.CatalogManager method*), 13
get() (*koordinates.exports.CropFeatureManager method*), 30
get() (*koordinates.exports.CropLayerManager method*), 29
get() (*koordinates.exports.ExportManager method*), 26
get() (*koordinates.layers.LayerManager method*), 15
get() (*koordinates.licenses.LicenseManager method*), 17
get() (*koordinates.permissions.PermissionManager method*), 31
get() (*koordinates.publishing.PublishManager method*), 19
get() (*koordinates.sets.SetManager method*), 20
get() (*koordinates.sources.SourceManager method*), 24
get() (*koordinates.users.GroupManager method*), 23
get() (*koordinates.users.UserManager method*), 22
get_creative_commons() (*koordinates.licenses.LicenseManager method*), 17
get_datasource() (*koordinates.sources.SourceManager method*), 24
get_draft() (*koordinates.layers.LayerManager method*), 15
get_draft() (*koordinates.sets.SetManager method*), 20
get_draft_version() (*koordinates.layers.Layer method*), 14
get_draft_version() (*koordinates.sets.Set method*), 21
get_feature() (*koordinates.exports.CropLayer method*), 29
get_feature() (*koordinates.exports.CropLayerManager method*), 29
get_formats() (*koordinates.exports.ExportManager method*), 26
get_formats() (*koordinates.metadata.Metadata method*), 18
get_items() (*koordinates.publishing.Publish method*), 19
get_log_lines() (*koordinates.sources.Scan method*), 25
get_manager() (*koordinates.client.Client method*), 12
get_published() (*koordinates.layers.LayerManager method*), 15
get_published() (*koordinates.sets.SetManager method*), 20
get_published_version() (*koordinates.layers.Layer method*), 14
get_published_version() (*koordinates.sets.Set method*), 21
get_scan() (*koordinates.sources.SourceManager method*), 24
get_scan_log_lines() (*koordinates.sources.SourceManager method*), 24
get_url() (*koordinates.client.Client method*), 12
get_version() (*koordinates.layers.Layer method*), 14
get_version() (*koordinates.layers.LayerManager method*), 15
get_version() (*koordinates.sets.Set method*), 21
get_version() (*koordinates.sets.SetManager method*), 20
get_xml() (*koordinates.metadata.Metadata method*), 18
Group (*class in koordinates.users*), 23
GroupManager (*class in koordinates.users*), 22

I

InternalServerError, 31
InvalidAPIVersion, 30
is_draft_version() (*koordinates.layers.Layer property*), 14

is_draft_version() (*koordinates.sets.Set property*), 21
is_published_version() (*koordinates.layers.Layer property*), 14
is_published_version() (*koordinates.sets.Set property*), 21

K

koordinates
    module, 12, 13, 17–19, 22, 23, 26, 31
koordinates.exceptions
    module, 30
KoordinatesException, 30

L

Layer (*class in koordinates.layers*), 13
LayerData (*class in koordinates.layers*), 16
LayerDataManager (*class in koordinates.layers*), 16
LayerManager (*class in koordinates.layers*), 15
LayerVersion (*class in koordinates.layers*), 16
LayerVersionManager (*class in koordinates.layers*), 16
License (*class in koordinates.licenses*), 17
LicenseManager (*class in koordinates.licenses*), 17
list() (*koordinates.catalog.CatalogManager method*), 13
list() (*koordinates.exports.CropFeatureManager method*), 30
list() (*koordinates.exports.CropLayerManager method*), 29
list() (*koordinates.exports.ExportManager method*), 27
list() (*koordinates.layers.LayerManager method*), 15
list() (*koordinates.licenses.LicenseManager method*), 17
list() (*koordinates.permissions.PermissionManager method*), 31
list() (*koordinates.publishing.PublishManager method*), 19
list() (*koordinates.sets.SetManager method*), 20
list() (*koordinates.sources.SourceManager method*), 24
list() (*koordinates.users.GroupManager method*), 23
list() (*koordinates.users.UserManager method*), 22
list_datasources() (*koordinates.sources.SourceManager method*), 24
list_drafts() (*koordinates.layers.LayerManager method*), 16
list_drafts() (*koordinates.sets.SetManager method*), 20
list_latest() (*koordinates.catalog.CatalogManager method*), 13
list_scans() (*koordinates.sources.SourceManager method*), 24
list_versions() (*koordinates.layers.Layer method*), 14
list_versions() (*koordinates.layers.LayerManager method*), 16
list_versions() (*koordinates.sets.Set method*), 21
list_versions() (*koordinates.sets.SetManager method*), 20

M

Metadata (*class in koordinates.metadata*), 18
MetadataManager (*class in koordinates.metadata*), 18
model (*koordinates.catalog.CatalogManager attribute*), 13
model (*koordinates.exports.CropFeatureManager attribute*), 30
model (*koordinates.exports.CropLayerManager attribute*), 29
model (*koordinates.exports.ExportManager attribute*), 27
model (*koordinates.layers.LayerDataManager attribute*), 16
model (*koordinates.layers.LayerManager attribute*), 16
model (*koordinates.layers.LayerVersionManager attribute*), 16
model (*koordinates.licenses.LicenseManager attribute*), 17
model (*koordinates.metadata.MetadataManager attribute*), 18
model (*koordinates.permissions.PermissionManager attribute*), 31
model (*koordinates.publishing.PublishManager attribute*), 19
model (*koordinates.sets.SetDataManager attribute*), 22
model (*koordinates.sets.SetManager attribute*), 20
model (*koordinates.sets.SetVersionManager attribute*), 22
model (*koordinates.sources.SourceManager attribute*), 24
model (*koordinates.users.GroupManager attribute*), 23
model (*koordinates.users.UserManager attribute*), 22
module
    koordinates, 12, 13, 17–19, 22, 23, 26, 31
    koordinates.exceptions, 30

N

NotAllowed, 31
NotFound, 30

O

order_by() (*koordinates.catalog.CatalogManager method*), 13

order_by() (*koordinates.exports.CropFeatureManager method*), 30
order_by() (*koordinates.exports.CropLayerManager method*), 29
order_by() (*koordinates.exports.ExportManager method*), 27
order_by() (*koordinates.layers.LayerManager method*), 16
order_by() (*koordinates.licenses.LicenseManager method*), 17
order_by() (*koordinates.publishing.PublishManager method*), 19
order_by() (*koordinates.sets.SetManager method*), 20
order_by() (*koordinates.sources.SourceManager method*), 24
order_by() (*koordinates.users.GroupManager method*), 23
order_by() (*koordinates.users.UserManager method*), 22

P

Permission (*class in koordinates.permissions*), 31
PermissionManager (*class in koordinates.permissions*), 31
Publish (*class in koordinates.publishing*), 19
publish() (*koordinates.layers.Layer method*), 14
publish() (*koordinates.sets.Set method*), 21
PublishManager (*class in koordinates.publishing*), 18

R

RateLimitExceeded, 31
refresh() (*koordinates.catalog.CatalogEntry method*), 12
refresh() (*koordinates.exports.CropFeature method*), 30
refresh() (*koordinates.exports.CropLayer method*), 29
refresh() (*koordinates.exports.Export method*), 28
refresh() (*koordinates.layers.Layer method*), 14
refresh() (*koordinates.licenses.License method*), 17
refresh() (*koordinates.publishing.Publish method*), 19
refresh() (*koordinates.sets.Set method*), 21
refresh() (*koordinates.sources.Datasource method*), 26
refresh() (*koordinates.sources.Scan method*), 26
refresh() (*koordinates.sources.Source method*), 24
refresh() (*koordinates.sources.UploadSource method*), 25
refresh() (*koordinates.users.Group method*), 23
refresh() (*koordinates.users.User method*), 22
reverse_url() (*koordinates.client.Client method*), 12

S

save() (*koordinates.layers.Layer method*), 14
save() (*koordinates.sets.Set method*), 21
save() (*koordinates.sources.Source method*), 25
save() (*koordinates.sources.UploadSource method*), 25
Scan (*class in koordinates.sources*), 25
ServerError, 30
ServiceUnvailable, 31
Set (*class in koordinates.sets*), 20
set() (*koordinates.metadata.MetadataManager method*), 18
set() (*koordinates.permissions.PermissionManager method*), 31
set_metadata() (*koordinates.layers.Layer method*), 14
set_metadata() (*koordinates.layers.LayerManager method*), 16
set_metadata() (*koordinates.sets.Set method*), 21
set_metadata() (*koordinates.sets.SetManager method*), 20
SetData (*class in koordinates.sets*), 22
SetDataManager (*class in koordinates.sets*), 22
SetManager (*class in koordinates.sets*), 19
SetVersion (*class in koordinates.sets*), 21
SetVersionManager (*class in koordinates.sets*), 22
Source (*class in koordinates.sources*), 24
SourceManager (*class in koordinates.sources*), 23
start_import() (*koordinates.layers.Layer method*), 14
start_import() (*koordinates.layers.LayerManager method*), 16
start_scan() (*koordinates.sources.SourceManager method*), 24
start_update() (*koordinates.layers.Layer method*), 15
start_update() (*koordinates.layers.LayerManager method*), 16

U

UploadSource (*class in koordinates.sources*), 25
User (*class in koordinates.users*), 22
UserManager (*class in koordinates.users*), 22

V

validate() (*koordinates.exports.ExportManager method*), 27