Isogeo Python SDK

Release 3.8.0

Isogeo

Dec 05, 2022

QUICKSTART

Author
: Julien M. (Isogeo)

Source code
: https://github.com/Isogeo/isogeo-api-py-minsdk/

Issues
: https://github.com/Isogeo/isogeo-api-py-minsdk/issues

Updated: 2022-12-05

CHAPTER ONE

INDICES AND TABLES

- genindex
- modindex
- search

1.1 Installation

```
pip install isogeo-pysdk
```

1.2 Authenticate to the API

1.2.1 Load API credentials from a JSON or INI file

Isogeo delivers API credentials in a JSON file. Its structure depends on the kind of oAuth2 application you are developing. Please referer to the API documentation to know more about different types of oAuth2 application.

For example, here is the JSON structure for a "workgroup" application:

```
{
"web": {
    "client_id": "python-minimalist-sdk-test-uuid-1a2b3c4d5e6f7g8h9i0j11k12l",
    "client_secret": "application-secret-1a2b3c4d5e6f7g8h9i0j11k12l13m14n15o16p17Q18rS",
    "auth_uri": "https://id.api.isogeo.com/oauth/authorize",
    "token_uri": "https://id.api.isogeo.com/oauth/token"
    }
}
```

The module isogeo_pysdk.utils comes with a method to load automatically credentials from JSON and INI files:

```
# load package
from isogeo_pysdk import Isogeo, IsogeoUtils

# instanciate IsogeoUtils as utils
utils = IsogeoUtils()

# load from file
api_credentials = utils.credentials_loader("client_secrets_group.json")
```

(continues on next page)

(continued from previous page)

```
# could also be:
# api_credentials = utils.credentials_loader("client_secrets_user.json")
# api_credentials = utils.credentials_loader("client_secrets.ini")

# authenticate your client application
isogeo = Isogeo(client_id=api_credentials.get("client_id"),
                client_secret=api_credentials.get("client_secret")
                )

# get the token
isogeo.connect()
```

Keys of returned dict:

- auth_mode
- client_id
- client_secret
- scopes
- uri_auth
- uri_base
- uri_redirect
- uri_token

1.2.2 Authenticate using oAuth2 Client Credentials Grant (”group application”)

This is the oAuth2 Backend Application flow, used by the named “group application” in Isogeo terms.

Supposing secrets are stored as environment variables:

```
# for oAuth2 Backend (Client Credentials Grant) Flow
isogeo = Isogeo(
    auth_mode="group",
    client_id=environ.get("ISOGEO_API_GROUP_CLIENT_ID"),
    client_secret=environ.get("ISOGEO_API_GROUP_CLIENT_SECRET"),
    auto_refresh_url="{}/oauth/token".format(environ.get("ISOGEO_ID_URL")),
    platform=environ.get("ISOGEO_PLATFORM", "qa"),
    max_retries=1,
)

# getting a token
isogeo.connect()
```

1.3 Usage

1.3.1 Manipulate shares

Shares are the features allowing to manage access to metadata catalogs via applications such as GIS plugins or Isogeo2office.

Get informations about shares (2 methods)

There are several ways to obtain more or less detailed informations about the shares accessible to an API client.

```python
from isogeo_pysdk import Isogeo

# authenticate your client application
isogeo = Isogeo(
    client_id=app_id,
    client_secret=app_secret,
    auto_refresh_url=isogeo_token_uri,
)

# get the token
isogeo.connect()

# -- BASIC INFORMATIONS ------------------------------------------
# by making a search
search = isogeo.search(page_size=0, whole_results=0, augment=1)  # set augment option on␣
↪True
# retieving shares uuid and name from search tags
shares_basic = ["{}:{}".format(k, v) for k, v in search.tags.items() if k.startswith(
↪'share:')]
print(shares_basic)

# -- DETAILED INFORMATIONS ---------------------------------------
# through API client attribute
shares_detailed = isogeo._shares
print(shares_detailed)

# properly closing connection
isogeo.close()
```

Get OpenCatalog URL for a share

OpenCatalog is an online metadata browser generated on Isogeo platform. As a third-party application, it can be set or not in a share.

Here is how to check if it's set or not in a share.

```python
from isogeo_pysdk import Isogeo

# authenticate your client application
isogeo = Isogeo(
```

(continues on next page)

(continued from previous page)

```
    client_id=app_id,
    client_secret=app_secret,
    auto_refresh_url=isogeo_token_uri
)

# get the token
isogeo.connect()

# -- RETRIEVE A SHARE UUID USING SEARCH ROUTE ------------------------------------------
# get a list of detailed informations about shares accessible by the API client
shares = isogeo.share.listing()

# get one share UUID in the list
share_id = shares[0].get("_id")

# -- ONCE SHARE UUID RETRIEVED ---------------------------------------------
# make an augmented share request
share_augmented = isogeo.share.get(share_id)

if share_augmented.opencatalog_url() is not None:
    print("OpenCatalog is set: {}".format(share_augmented.opencatalog_url()))
else:
    print("OpenCatalog is not set in this share")
```

Check if a metadata is in a share

With the augmented share, it's also possible to check if a metadata is present within.

```
# -- see above to get augmented share

# get a metadata
search = isogeo.search(
    page_size=1,     # get only one result
    whole_results=0    # do not retrieve the whole application scope
)
# retrieve metadata uuid
md_id = md_search.results[0].get("_id")

# make a search on metadatas accessibles through the share
share_search = isogeo.search(
    share=share_augmented.get("_id"),  # filter on the share
    whole_results=1  # retrieve the whole application scope
)
# check if the metadata is in the result
share_mds_ids = [md.get("_id") for md in share_search.results]

if md_id in share_mds_ids:
    print("Metadata is present in this share.")
else:
    print("Not present.")
```

(continues on next page)

(continued from previous page)

```python
# properly closing connection
isogeo.close()
```

1.3.2 URL Builder for web applications

All examples in this section must be precede by the following code:

```python

from isogeo_pysdk import IsogeoUtils, Isogeo

utils = IsogeoUtils()
# authenticate your client application
isogeo = Isogeo(
    client_id=app_id,
    client_secret=app_secret,
    auto_refresh_url=isogeo_token_uri
)

# get the token
isogeo.connect()
```

Isogeo metadata can be displyed in others web applications. Some webapps are built-in:

- OpenCatalog (oc)
- *Data portal by PixUp (pixup_portal)
- *CSW GetCapabilities (for a share)
- *CSW GetRecords (for a metadata)

It's also possible to register a custom web app (see below).

Get URL to online editor for a metadata

A metadata can only be edited by an authenticated Isogeo user (with editor level at least). A built-in method make it easy to contruct it:

```python
md = isogeo.metadata.get(md_id="36fde4261bcb4ef2a849d94a50488713")

url = utils.get_edit_url(metadata=md, tab="attributes")
```

Get OpenCatalog URL for a metadata

```python
oc_args = {
    "md_id": "36fde4261bcb4ef2a849d94a50488713",
    "share_id": "344d51c3edfb435daf9d98d948fa207e",
    "share_token": "TokenOhDearToken"
}

oc_url = utils.get_view_url(webapp="oc", **oc_args)
```

Get CSW GetCapabilities for a share

```python
csw_getcap_args = {
    "share_id": "344d51c3edfb435daf9d98d948fa207e",
    "share_token": "TokenOhDearToken",
}

csw_getcap_url = utils.get_view_url(webapp="csw_getcap", **csw_getcap_args)
```

Get CSW GetRecords for a share

```python
csw_getrecords_args = {
    "share_id": "344d51c3edfb435daf9d98d948fa207e",
    "share_token": "TokenOhDearToken",
}

csw_getrecords_url = utils.get_view_url(webapp="csw_getrecords", **csw_getrecords_args)
```

Get CSW GetRecordById for a metadata

```python
uuid_md_source = "36fde4261bcb4ef2a849d94a50488713"

csw_getrec_args = {
    "md_uuid_urn": utils.convert_uuid(uuid_md_source, 2),
    "share_id": "344d51c3edfb435daf9d98d948fa207e",
    "share_token": "TokenOhDearToken"
}

csw_getrec_url = utils.get_view_url(webapp="csw_getrec", **csw_getrec_args)
```

Register a custom webapp and get URL

```python
# register the web app
utils.register_webapp(
    webapp_name="PPIGE v3",
    webapp_args=["md_id", ],
    webapp_url="https://www.ppige-npdc.fr/portail/geocatalogue?uuid={md_id}"
)
# get url
custom_url_args = {
    "md_id": "36fde4261bcb4ef2a849d94a50488713",
    "share_id": "344d51c3edfb435daf9d98d948fa207e",
    "share_token": "TokenOhDearToken"
}
custom_url = utils.get_view_url(webapp="PPIGE v3", **custom_url_args)
```

1.3.3 Download metadata as XML ISO 19139

In Isogeo, every metadata resource can be downloaded in its XML version (ISO 19139 compliant). The Python SDK package inclue a shortcut method:

```python

from isogeo_pysdk import Isogeo, Metadata

# authenticate your client application
isogeo = Isogeo(
    client_id=app_id,
    client_secret=app_secret,
    auto_refresh_url=isogeo_token_uri
)

# get the token
isogeo.connect()

# search metadata
search_to_be_exported = isogeo.search(
    page_size=10,
    query="type:dataset",
    whole_results=0
)

# loop on results and export
for md in search_to_be_exported.results:
    metadata = Metadata.clean_attributes(md)
    title = metadata.title
    xml_stream = isogeo.metadata.download_xml(metadata)

    with open("{}.xml".format(title), 'wb') as fd:
        for block in xml_stream.iter_content(1024):
            fd.write(block)
```

(continues on next page)

(continued from previous page)

```python
# properly closing connection
isogeo.close()
```

Others examples:

- Batch export into XML within Isogeo to Office application.
- Batch export into XML in the package sample.

1.3.4 Download hosted data from Isogeo cloud

Administrators and editors can link raw data and docs (.zip, .pdf…) to metadata to allow final users to access the data. To do that, it's possible to upload data to Isogeo cloud (Azure blob storage).Through the API, it's possible to download these data:

```python
from isogeo_pysdk import Isogeo

# authenticate your client application
isogeo = Isogeo(
    client_id=app_id,
    client_secret=app_secret,
    auto_refresh_url=isogeo_token_uri
)

# get the token
isogeo.connect()

# search with _include = links and action = download
latest_data_modified = isogeo.search(
    page_size=10,
    order_by="modified",
    whole_results=0,
    query="action:download",
    include=("links",),
)

# parse links and download hosted data recursively
for md in latest_data_modified.results:
    for link in filter(lambda x: x.get("type") == "hosted", md.get("links")):
        dl_stream = isogeo.metadata.links.download_hosted(link=link)
        filename = re.sub(r'[\\/*?:"<>|]', "", dl_stream[1])
        with open(dl_stream[1], "wb") as fd:
            for block in dl_stream[0].iter_content(1024):
                fd.write(block)

# properly closing connection
isogeo.close()
```

Example:

- Batch export hosted data in the package sample.

1.3.5 Add versions to a format

It is necessary to update Isogeo database with new formats versions, so that users can properly fill metadata sheets on app.isogeo.com.

```python
from isogeo_pysdk import Isogeo

# -- Authentication using 'user_legacy' flow
# Isogeo client
isogeo = Isogeo(
    auth_mode="user_legacy",
    client_id=api_dev_id,
    client_secret=api_dev_secret,
    auto_refresh_url=isogeo_token_uri
)

# getting a token
isogeo.connect(
    username=isogeo_user_name,
    password=isogeo_user_password,
)

# get a format
fmt_postgis = isogeo.formats.get("postgis")
# add versions to this format
fmt_postgis.versions.extend(li_new_versions)
# update the format with added versions
fmt_postgis_updated = isogeo.formats.update(fmt_postgis)

# properly closing connection
isogeo.close()
```

1.4 isogeo_pysdk

1.4.1 isogeo_pysdk package

This module is an abstraction class about the Isogeo REST API.

https://www.isogeo.com/

Subpackages

isogeo_pysdk.api package

Submodules

isogeo_pysdk.api.routes_about module

Module to easily get versions about Isogeo platform components.

`class isogeo_pysdk.api.routes_about.`**`ApiAbout`**`(`*platform='prod'*, *proxies=None*, *protocol='https'*, *isogeo_urls={}*`)`

Bases: `object`

Routes as methods of Isogeo API used to get platform informations.

`api``()`

Get API version.

Return type
`str`

`authentication``()`

Get authentication server (ID) version.

Return type
`str`

`database``()`

Get database version.

Return type
`str`

`scan``()`

Get daemon version.

Return type
`str`

`services``()`

Get services.api version.

Return type
`str`

isogeo_pysdk.api.routes_account module

Isogeo API v1 - API Routes for Account entities

See: http://help.isogeo.com/api/complete/index.html

`class isogeo_pysdk.api.routes_account.`**`ApiAccount`**`(`*api_client=None*`)`

Bases: `object`

Routes as methods of Isogeo API used to manipulate account (user).

get(*include=('_abilities',), caching=1*)

Get authenticated user account(= profile) informations.

Parameters

- **include** (*tuple*) – additional parts of model to include in response
- **caching** (*bool*) – option to cache the response

Return type

User

memberships()

Returns memberships for the authenticated user.

Example

```
>>> my_groups = isogeo.account.memberships()
>>> print(len(my_groups))
10
>>> groups_where_iam_admin = list(filter(lambda d: d.get("role") == "admin", my_
↪groups))
>>> print(len(groups_where_iam_admin))
5
>>> groups_where_iam_editor = list(filter(lambda d: d.get("role") == "editor",␣
↪my_groups))
>>> print(len(groups_where_iam_editor))
4
>>> groups_where_iam_reader = list(filter(lambda d: d.get("role") == "reader",␣
↪my_groups))
>>> print(len(groups_where_iam_reader))
1
```

Return type

list

update(*account, caching=1*)

Update authenticated user account(= profile) informations.

Parameters

- **account** (*class*) – user account model object to update
- **caching** (*bool*) – option to cache the response

Return type

User

isogeo_pysdk.api.routes_application module

Isogeo API v1 - API Routes for Applications entities

See: http://help.isogeo.com/api/complete/index.html

`class isogeo_pysdk.api.routes_application.`**`ApiApplication`**`(`*api_client=None*`)`

Bases: `object`

Routes as methods of Isogeo API used to manipulate applications.

`associate_group``(`*application*, *workgroup*, *force=0*`)`

Associate a application with a workgroup.

Parameters

- **`application`** (`Application`) – Application model object to update
- **`workgroup`** (`Workgroup`) – object to associate
- **`force`** (*`bool`*) – option to force association with multiple groups changing the *canHave-ManyGroups* property

Return type

`tuple`

`create``(`*application*, *check_exists=1*`)`

Add a new application to Isogeo.

Parameters

`check_exists` (*`int`*) – check if a application already exists inot the workgroup:

- 0 = no check
- 1 = compare name [DEFAULT]

Parameters

`application` (*`class`*) – Application model object to create

Return type

`Application`

`delete``(`*application_id*`)`

Delete a application from Isogeo database.

Parameters

`application_id` (*`str`*) – identifier of the resource to delete

`dissociate_group``(`*application*, *workgroup*`)`

Removes the association between the specified group and the specified application.

Parameters

- **`application`** (`Application`) – Application model object to update
- **`workgroup`** (`Workgroup`) – object to associate

Return type

`tuple`

`exists`(*application_id*)

Check if the specified application exists and is available for the authenticated user.

Parameters
application_id (*str*) – identifier of the application to verify

Return type
bool

`get`(*application_id*, *include=('_abilities', 'groups')*)

Get details about a specific application.

Parameters

- **application_id** (*str*) – application UUID
- **include** (*tuple*) – additionnal subresource to include in the response

Return type
Application

`listing`(*workgroup_id=None*, *include=('_abilities',)*, *caching=1*)

Get all applications which are accessible by the authenticated user OR applications for a workgroup.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup. If *None*, then list applications for the autenticated user
- **include** (*tuple*) – additionnal subresource to include in the response.
- **caching** (*bool*) – option to cache the response

Return type
list

`update`(*application*, *caching=1*)

Update a application owned by a workgroup.

Parameters

- **application** (*class*) – Application model object to update
- **caching** (*bool*) – option to cache the response

Return type
Application

`workgroups`(*application_id=None*)

Get all groups associated with an application.

Parameters
application_id (*str*) – identifier of the application

Return type
list

isogeo_pysdk.api.routes_catalog module

Isogeo API v1 - API Routes for Catalogs entities

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.api.routes_catalog.**ApiCatalog**(*api_client=None*)

Bases: `object`

Routes as methods of Isogeo API used to manipulate catalogs.

associate_metadata(*metadata*, *catalog*)

Associate a metadata with a catalog.

If the specified catalog is already associated, the response is still 204.

Parameters

- **metadata** (`Union`[*Metadata*, `Tuple`[*Metadata*, `...`]]) – metadata object to update
- **catalog** (`Catalog`) – catalog model object to associate

Example

```
isogeo.catalog.associate_metadata(
    isogeo.metadata.get(METADATA_UUID),
    isogeo.catalog.get(WORKGROUP_UUID, CATALOG_UUID)
    )

<Response [204]>
```

Return type
`Response`

create(*workgroup_id*, *catalog*, *check_exists=1*)

Add a new catalog to a workgroup.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **catalog** (*class*) – Catalog model object to create
- **check_exists** (*bool*) – check if a catalog already exists into the workgroup:
  - 0 = no check
  - 1 = compare name [DEFAULT]

Returns
the created catalog or False if a similar cataog already exists or a tuple with response error code

Return type
Catalog

delete(*workgroup_id*, *catalog_id*)

Delete a catalog from Isogeo database.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup

- **catalog_id** (*str*) – identifier of the resource to delete

dissociate_metadata(*metadata*, *catalog*)

Removes the association between a metadata and a catalog.

If the specified catalog is not associated, the response is 404.

Parameters

- **metadata** (Metadata) – metadata object to update
- **catalog** (Catalog) – catalog model object to associate

Return type
Response

exists(*workgroup_id*, *catalog_id*)

Check if the specified catalog exists and is available for the authenticated user.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **catalog_id** (*str*) – identifier of the catalog to verify

Return type
bool

get(*workgroup_id*, *catalog_id*, *include=('_abilities', 'count')*)

Get details about a specific catalog.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **catalog_id** (*str*) – catalog UUID
- **include** (*list*) – additionnal subresource to include in the response

Return type
Catalog

listing(*workgroup_id=None*, *include=('_abilities', 'count')*, *caching=1*)

Get workgroup catalogs.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **include** (*tuple*) – additionnal subresource to include in the response
- **caching** (*bool*) – option to cache the response

Return type
list

Example

```
# retrieve the catalogs of workgroup
wg_catalogs = isogeo.catalog.listing(
    workgroup_id=isogeo_workgroup._id,
    include=None
)
# filter on catalogs with the Sacn checked
```

(continues on next page)

(continued from previous page)

```
for cat in wg_catalogs:
    if cat.get("$scan", False):
        print(cat.get("name"))
```

metadata(*metadata_id*)

List metadata's catalogs with complete information.

Parameters
metadata_id (*str*) – metadata UUID

Returns
the list of catalogs associated with the metadata

Return type
list

shares(*catalog_id*)

Returns shares for the specified catalog.

Parameters
catalog_id (*str*) – catalog UUID

Returns
list of Shares containing the catalog

Return type
list

statistics(*catalog_id*)

Returns statistics for the specified catalog.

Parameters
catalog_id (*str*) – catalog UUID

Return type
dict

statistics_by_tag(*catalog_id*, *tag*)

Returns statistics on a specific tag for the specified catalog.

Be careful: if an invalid character is present into the response (e.g. contact.name = 'bureau GF-3A'), a ConnectionError / ReadTimeout will be raised.

Parameters

- **catalog_id** (*str*) – catalog UUID
- **tag** (*str*) – tag name. Must be one of: catalog, coordinate-system, format, keyword:inspire-theme, keyword, owner

Return type
dict

update(*catalog*, *caching=1*)

Update a catalog owned by a workgroup.

Parameters

- **catalog** (*class*) – Catalog model object to update
- **caching** (*bool*) – option to cache the response

Return type
Catalog

isogeo_pysdk.api.routes_condition module

Isogeo API v1 - API Routes for Conditions entities

See: http://help.isogeo.com/api/complete/index.html

`class isogeo_pysdk.api.routes_condition.`**`ApiCondition`**`(`*api_client=None*`)`

Bases: `object`

Routes as methods of Isogeo API used to manipulate conditions.

`create``(`*metadata*, *condition*`)`

Add a new condition (license + specific description) to a metadata.

Parameters

- **`metadata`** (`Metadata`) – metadata object to update
- **`condition`** (`Condition`) – condition to create

Return type
Condition

`delete``(`*metadata*, *condition*`)`

Removes a condition from a metadata.

Parameters

- **`metadata`** (`Metadata`) – metadata object to update
- **`condition`** (`Condition`) – license model object to associate

Return type
`Response`

`get``(`*metadata_id*, *condition_id*`)`

Get details about a specific condition.

Parameters

- **`metadata_id`** (*str*) – identifier of the owner workgroup
- **`condition_id`** (*str*) – condition UUID

Return type
Condition

`listing``(`*metadata_id*`)`

List metadata's conditions with complete information.

Parameters
`metadata_id` (*str*) – metadata UUID

Returns
the list of conditions associated with the metadata

Return type
list

isogeo_pysdk.api.routes_conformity module

Isogeo API v1 - API Routes for Conformity entities

See: http://help.isogeo.com/api/complete/index.html

`class isogeo_pysdk.api.routes_conformity.`**`ApiConformity`**`(`*api_client=None*`)`

Bases: `object`

Routes as methods of Isogeo API used to manipulate conformity with specifications.

`create``(`*metadata*, *conformity*`)`

Add a new conformity (specification + specific conformant) to a metadata.

Parameters

- **`metadata`** (`Metadata`) – metadata object to update
- **`conformity`** (`Conformity`) – conformity to create

Return type

`Conformity`

`delete``(`*metadata*, *conformity=None*, *specification_id=None*`)`

Removes a conformity from a metadata.

Parameters

- **`metadata`** (`Metadata`) – metadata object to update
- **`conformity`** (`Conformity`) – specification model object to associate. If empty, the specification_id must be passed.
- **`specification_id`** (`Specification`) – specification model object to associate. If empty, the conformity must be passed.

Return type

`Response`

`listing``(`*metadata_id*`)`

List metadata's conformity specifications with complete information.

Parameters

`metadata_id` (*`str`*) – metadata UUID

Returns

the list of specifications + conformity status associated with the metadata

Return type

list

isogeo_pysdk.api.routes_contact module

Isogeo API v1 - API Routes for Contacts entities

See: http://help.isogeo.com/api/complete/index.html

`class isogeo_pysdk.api.routes_contact.`**`ApiContact`**`(`*api_client=None*`)`

Bases: `object`

Routes as methods of Isogeo API used to manipulate contacts.

associate_metadata(*metadata*, *contact*, *role='pointOfContact'*)

Associate a metadata with a contact.

If the specified contact is already associated, the response is still 200.

Parameters

- **metadata** (Metadata) – metadata object to update
- **contact** (Contact) – contact model object to associate
- **role** (*str*) – role to assign to the contact

Example

```
# retrieve a metadata
md = isogeo.metadata.get(METADATA_UUID)
# retrieve a contact
ctct = isogeo.contact.get(CONTACT_UUID)
# associate a contact to a metadata
isogeo.contact.associate_metadata(metadata = md, contact = ctct)
```

Return type

Response

create(*workgroup_id*, *contact*, *check_exists=1*)

Add a new contact to a workgroup.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **contact** (*class*) – Contact model object to create
- **check_exists** (*int*) – check if a contact already exists inot the workgroup:
  - 0 = no check
  - 1 = compare name [DEFAULT]
  - 2 = compare email

Returns

the created contact or the existing contact if case oof a matching name or email or a tuple with response error code

Return type

Contact

delete(*workgroup_id*, *contact_id*)

Delete a contact from Isogeo database.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **contact_id** (*str*) – identifier of the resource to delete

dissociate_metadata(*metadata*, *contact*)

Removes the association between a metadata and a contact.

If the specified contact is not associated, the response is 404.

Parameters

- **metadata** (Metadata) – metadata object to update
- **contact** (Contact) – contact model object to associate

Return type
Response

exists(*contact_id*)

Check if the specified contact exists and is available for the authenticated user.

Parameters
contact_id (*str*) – identifier of the contact to verify

Return type
bool

get(*contact_id*)

Get details about a specific contact.

Parameters
contact_id (*str*) – contact UUID

Return type
Contact

listing(*workgroup_id=None*, *include=('count',)*, *caching=1*)

Get workgroup contacts.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **include** (*tuple*) – identifier of the owner workgroup
- **caching** (*bool*) – option to cache the response

Return type
list

update(*contact*, *caching=1*)

Update a contact owned by a workgroup.

Parameters

- **contact** (*class*) – Contact model object to update
- **caching** (*bool*) – option to cache the response

Return type
Contact

isogeo_pysdk.api.routes_coordinate_systems module

Isogeo API v1 - API Routes to retrieve CoordinateSystems

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.api.routes_coordinate_systems.**ApiCoordinateSystem**(*api_client=None*)

Bases: object

Routes as methods of Isogeo API used to manipulate coordinate-systems.

associate_metadata(*metadata*, *coordinate_system*)

Associate a coordinate-system (SRS) to a metadata.

If a coordinate-system is already associated to the metadata, it'll be oversritten.

Parameters

- **metadata** (Metadata) – metadata object to update
- **coordinate_system** (CoordinateSystem) – coordinate-system model object to associate

Return type

CoordinateSystem

Example

```
# retrieve metadata
md = isogeo.metadata.get(
        metadata_id=METADATA_UUID,
        include=()
)
# retrieve one of the SRS selected in the workgroup of the metadata
wg_srs = self.isogeo.coordinate_system.listing(md._creator.get("_id"))
random_srs = CoordinateSystem(**sample(wg_srs, 1)[0])
# associate them
isogeo.coordinateSystem.associate_metadata(
    metadata=md,
    coordinateSystem=random_srs,
)
```

associate_workgroup(*coordinate_system*, *workgroup=None*)

Add a coordinate system to the workgroup selection or/adn edit the SRS custom alias.

Parameters

- **coordinate_system** (CoordinateSystem) – EPSG code of the coordinate system to add to the workgroup selection
- **workgroup** (Workgroup) – identifier of the owner workgroup.

Return type

CoordinateSystem

Example

```
# retrieve the SRS
coordsys = isogeo.srs.get("4326")
# add a custom alias
coordsys.alias = "World SRS"
# add it to the workgroup selection
isogeo.srs.associate_workgroup(
    workgroup=isogeo.workgroup.get(WORKGROUP_UUID),
    coordinate_system=coordsys
)
```

dissociate_metadata(*metadata*)

Removes the coordinate-system from a metadata.

Parameters
`metadata` (`Metadata`) – metadata object to update

Return type
`Response`

`dissociate_workgroup`(*coordinate_system_code*, *workgroup_id=None*)

Remove a coordinate system from the workgroup selection.

Parameters

- `coordinate_system_code` (*str*) – EPSG code of the coordinate system to reomve from the workgroup selection
- `workgroup_id` (*str*) – identifier of the owner workgroup.

Return type
CoordinateSystem

Example

```
>>> isogeo.srs.dissociate_workgroup(
    workgroup_id=WORKGROUP_TEST_FIXTURE_UUID,
    coordinate_system_code="2154"
    )
```

`get`(*coordinate_system_code*, *workgroup_id=None*)

Get details about a specific coordinate_system, from the whole Isogeo database or into a specific workgroup (to get the SRS alias for example).

Parameters

- `workgroup_id` (*str*) – identifier of the owner workgroup. OPTIONNAL: if present, list SRS slected into the workgroup.
- `coordinate_system_code` (*str*) – EPSG code of the coordinate system

Return type
CoordinateSystem

Example

```
>>> # list all coordinate-systems in the whole Isogeo database
>>> srs = isogeo.srs.listing()
>>> # print details about the first SRS found
>>> pprint.pprint(isogeo.srs.get(srs[0].get("code")))
{
    '_tag': 'coordinate-system:4143',
    'code': 4143,
    'name': 'Abidjan 1987'
}
```

`listing`(*workgroup_id=None*, *caching=1*)

Get coordinate-systems in the whole Isogeo database or into a specific workgroup.

Parameters

- `workgroup_id` (*str*) – identifier of the owner workgroup. OPTIONNAL: if present, list SRS slected into the workgroup.
- `caching` (*bool*) – option to cache the response

Return type
list

Example

```
>>> # list all coordinate-systems in the whole Isogeo database
>>> srs = isogeo.srs.listing()
>>> print(len(srs))
4301
>>> # list coordinate-systems which have been selected in a specific workgroup
>>> srs = isogeo.srs.listing(workgroup_id=WORKGROUP_UUID)
>>> print(len(srs))
5
```

isogeo_pysdk.api.routes_datasource module

Isogeo API v1 - API Routes for Datasources entities

See: http://help.isogeo.com/api/complete/index.html

`class isogeo_pysdk.api.routes_datasource.`**`ApiDatasource`**(*api_client=None*)

Bases: `object`

Routes as methods of Isogeo API used to manipulate datasources (CSW entry-points).

`create`(*workgroup_id, datasource={'_created': None, '_id': None, '_modified': None, '_tag': None, 'enabled': None, 'lastSession': None, 'location': None, 'name': None, 'resourceCount': None, 'sessions': None}, check_exists=2*)

Add a new datasource to a workgroup.

Parameters

- **`workgroup_id`** (*`str`*) – identifier of the owner workgroup
- **`check_exists`** (*`int`*) – check if a datasource already exists inot the workgroup:

- 0 = no check
- 1 = compare name
- 2 = compare URL (location) [DEFAULT]

Parameters
`datasource` (*`class`*) – Datasource model object to create

Return type
`Datasource`

`datasource`(*workgroup_id, datasource_id*)

Get details about a specific datasource.

Parameters

- **`workgroup_id`** (*`str`*) – identifier of the owner workgroup
- **`datasource_id`** (*`str`*) – datasource UUID

Return type
`Datasource`

delete(*workgroup_id*, *datasource_id*)

Delete a datasource from Isogeo database.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **datasource_id** (*str*) – identifier of the resource to delete

exists(*workgroup_id*, *datasource_id*)

Check if the specified datasource exists and is available for the authenticated user.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **datasource_id** (*str*) – identifier of the datasource to verify

Return type

`bool`

listing(*workgroup_id=None*, *include=None*, *caching=1*)

Get workgroup datasources.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **include** (*list*) – additionnal subresource to include in the response
- **caching** (*bool*) – option to cache the response

Return type

`list`

update(*workgroup_id*, *datasource*, *caching=1*)

Update a datasource owned by a workgroup.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **datasource** (*class*) – Datasource model object to update
- **caching** (*bool*) – option to cache the response

Return type

Datasource

isogeo_pysdk.api.routes_directives module

Isogeo API v1 - API Routes to retrieve EU environment code Directives used as INSPIRE limitations

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.api.routes_directives.**ApiDirective**(*api_client=None*)

Bases: `object`

Routes as methods of Isogeo API used to manipulate directives (Europe Environment code for INSPIRE limitations).

listing(*caching=1*)

Get directives.

Parameters
caching (*bool*) – option to cache the response

Return type
list

isogeo_pysdk.api.routes_event module

Isogeo API v1 - API Routes for Events entities

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.api.routes_event.**ApiEvent**(*api_client=None*)

Bases: object

Routes as methods of Isogeo API used to manipulate events.

create(*metadata*, *event*)

Add a new event to a metadata (= resource).

Parameters

- **metadata** (Metadata) – metadata (resource) to edit
- **Event** (Event) – event object to create

Return type
Event

delete(*event*, *metadata=None*)

Delete a event from Isogeo database.

Parameters

- **event** (*class*) – Event model object to delete
- **metadata** (Metadata) – parent metadata (resource) containing the event

event(*metadata_id*, *event_id*)

Get details about a specific event.

Parameters

- **event_id** (*str*) – event UUID to get
- **event_id** – event UUID

Return type
Event

listing(*metadata*)

Get all events of a metadata.

Parameters
metadata (Metadata) – metadata (resource) to edit

Return type
list

update(*event*, *metadata=None*)

Update an event.

Parameters

- **event** (*class*) – Event model object to update
- **metadata** (Metadata) – parent metadata (resource) containing the event

Return type

Event

isogeo_pysdk.api.routes_feature_attributes module

Isogeo API v1 - API Routes for FeatureAttributes entities

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.api.routes_feature_attributes.**ApiFeatureAttribute**(*api_client=None*)

Bases: object

Routes as methods of Isogeo API used to manipulate feature attributes into a Metadata.

create(*metadata*, *attribute*)

Add a new feature attribute to a metadata (= resource).

Parameters

- **metadata** (Metadata) – metadata (resource) to edit
- **attribute** (FeatureAttribute) – feature-attribute object to create

Returns

409 if an attribute with the same name already exists.

Return type

FeatureAttribute or tuple

Example

```
>>> # retrieve metadata
>>> md = isogeo.metadata.get(METADATA_UUID)
>>> # create the attribute locally
>>> new_attribute = FeatureAttribute(
    alias="Code INSEE de la commune",
    name="INSEE_COM",
    description="Une commune nouvelle résultant d’un regroupement de communes "
    "préexistantes se voit attribuer  le  code  INSEE  de  l’ancienne  commune "
    "désignée  comme  chef-lieu  par  l’arrêté  préfectoral  qui  l’institue. "
    "En  conséquence  une  commune change  de  code  INSEE  si  un  arrêté "
    "préfectoral modifie son chef-lieu.",
    dataType="CharacterString (5)",
    language="fr",
    )
>>> # add it to the metadata
>>> isogeo.metadata.attributes.create(md, new_attribute)
```

delete(*attribute*, *metadata=None*)

Delete a feature-attribute from a metadata.

Parameters

- **attribute** (FeatureAttribute) – FeatureAttribute model object to delete
- **metadata** (Metadata) – parent metadata (resource) containing the feature-attribute

get(*metadata_id*, *attribute_id*)

Get details about a specific feature-attribute.

Parameters

- **metadata_id** (*str*) – metadata UUID
- **attribute_id** (*str*) – feature-attribute UUID

Example

```
>>> # get a metadata
>>> md = isogeo.metadata.get(METADATA_UUID)
>>> # list all og its attributes
>>> md_attributes = isogeo.metadata.attributes.listing(md)
>>> # get the first attribute
>>> attribute = isogeo.metadata.attributes.get(
    metadata_id=md._id,
    attribute_id=md_attributes[0].get("_id")
    )
```

Return type

FeatureAttribute

import_from_dataset(*metadata_source*, *metadata_dest*, *mode='add'*, *case_sensitive_matching=True*)

Import feature-attributes from another vector (or DTNG) metadata.

Parameters

- **metadata_source** (Metadata) – metadata from which to import the attributes
- **metadata_dest** (Metadata) – metadata where to import the attributes
- **mode** (*str*) – mode of import, defaults to 'add':
  - 'add': add the attributes except those with a duplicated name
  - 'update': update only the attributes with the same name
  - 'update_or_add': update the attributes with the same name or create

: param bool case_sensitive_matching: False to make featureattributes's name matching case-insensitive when mode == "update"

Raises

- **TypeError** – if one metadata is not a vector or DTNG
- **ValueError** – if mode is not one of accepted value

Example

```
# get the metadata objects
md_source = isogeo.metadata.get(METADATA_UUID_SOURCE)
md_dest = isogeo.metadata.get(METADATA_UUID_DEST)

```

(continues on next page)

(continued from previous page)

```
# launch import
isogeo.metadata.attributes.import_from_dataset(md_source, md_dest, "add")
```

Return type
bool

listing(*metadata*)

Get all feature-attributes of a metadata.

Parameters
metadata (Metadata) – metadata (resource)

Return type
list

update(*attribute*, *metadata=None*)

Update a feature-attribute.

Parameters

- **attribute** (FeatureAttribute) – Feature Attribute model object to update
- **metadata** (Metadata) – parent metadata (resource) containing the feature-attribute

Return type
FeatureAttribute

isogeo_pysdk.api.routes_format module

Isogeo API v1 - API Routes for Formats entities

See: http://help.isogeo.com/api/complete/index.html

NOTE: *format* being the name of a Python built-in function (see: https://docs.python.org/3/library/functions.html#format), we use the *frmt* shorter as replacement.

class isogeo_pysdk.api.routes_format.**ApiFormat**(*api_client=None*)

Bases: object

Routes as methods of Isogeo API used to manipulate formats.

create(*frmt*, *check_exists=1*)

Add a new format to the Isogeo formats database.

If a format with the same code already exists, the Isogeo API returns a 500 HTTP code. To prevent this error, use the check_exists option or check by yourself before.

Parameters

- **frmt** (Format) – Format model object to create
- **check_exists** (*bool*) – check if a format with the same code exists before

Return type
Format or tuple

Example

```
format_to_add = Format(
    code="geojson",
    name="GeoJSON",
    type="vectorDataset"
)
isogeo.formats.create(format_to_add)
```

delete(*frmt*)

Delete a format from Isogeo database.

Parameters
frmt (Format) – Format model object to delete

get(*format_code*)

Get details about a specific format.

Parameters
format_code (*str*) – format code

Return type
Format

Example

```
>>> pprint.pprint(isogeo.formats.get("postgis"))
{
    '_id': string (uuid),
    '_tag': 'format:postgis',
    'aliases': [],
    'code': 'postgis',
    'name': 'PostGIS',
    'type': 'dataset',
    'versions': [
        '2.2',
        '2.1',
        '2.0',
        '1.5',
        '1.4',
        '1.3',
        '1.2',
        '1.1',
        '1.0',
        '0.9',
        None
        ]
}
```

listing(*data_type=None*, *caching=1*)

List formats available in Isogeo API.

Parameters

- **data_type** (*str*) – type of metadata to filter on
- **caching** (*bool*) – option to cache the response

Returns
list of dicts

Return type
list

Example

```
>>> formats = isogeo.formats.listing()
>>> # count all formats
>>> print(len(formats))
32
>>> # count formats which are only for vector dataset
>>> print(len(isogeo.formats.listing(data_type="vector-dataset")))
21
>>> # list all unique codes
>>> formats_codes = [i.get("code") for i in formats]
>>> pprint.pprint(formats_codes)
[
    'apic',
    'arcsde',
    'dgn',
    'dwg',
    'dxf',
    ...
]
```

nogeo_search(*query=None*, *page_size=10*, *offset=0*)

Search within data formats available in Isogeo API for NON GEOGRAPHICAL DATA ONLY.

Parameters

- **query** (*str*) – search terms. Equivalent of **q** parameter in Isogeo API.
- **page_size** (*int*) – limits the number of results. Useful to paginate results display. Default value: 10. Max value: 100.
- **offset** (*int*) – offset to start page size from a specific results index

Returns
list of dicts

Return type
list

Example

```
>>> isogeo.formats.search(query="a", page_size=1, offset=0)
[
    {
        'aliases': [],
        'name': 'Adobe PDF',
        'versions':
        [
            '7',
            '1.7',
            None,
            None
        ]
    }
]
```

update(*frmt*)

Update a format in Isogeo database.

Parameters

frmt (Format) – Format model object to update

Return type

Format

Example

```
>>> # retrieve format to update
>>> fmt_postgis = isogeo.formats.get("postgis")
>>> # add new versions locally
>>> fmt_postgis.versions.extend(["3.0", "3.1"])
>>> # update online
>>> fmt_postgis_updted = isogeo.formats.update(fmt_postgis)
```

isogeo_pysdk.api.routes_invitation module

Isogeo API v1 - API Routes for Invitations entities

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.api.routes_invitation.**ApiInvitation**(*api_client=None*)

Bases: object

Routes as methods of Isogeo API used to manipulate invitations.

accept(*invitation=<class 'isogeo_pysdk.models.invitation.Invitation'>*)

Accept the invitation to join an Isogeo Workgroup.

Parameters

invitation (*class*) – Invitation model object to accept

Return type

Invitation

create(*workgroup_id, invitation={'_created': None, '_id': None, '_modified': None, 'email': None, 'expiresIn': None, 'group': None, 'role': None}*)

Add a new invitation to Isogeo.

Parameters

invitation (*class*) – Invitation model object to create

Return type

Invitation

Example

```
>>> # create the invitation locally
>>> invit = Invitation(
    email="prenom.nom@organisation.com",
    role="admin"
    )
>>> # send the invitation
>>> isogeo.invitation.create(WORKGROUP_UUID, new_invit)
```

decline(*invitation=<class 'isogeo_pysdk.models.invitation.Invitation'>*)

Decline the invitation to join an Isogeo Workgroup.

Parameters
invitation (*class*) – Invitation model object to decline

Return type
Invitation

delete(*invitation_id*)

Delete an invitation from Isogeo database.

Parameters
invitation_id (*str*) – identifier of the invitation

get(*invitation_id*)

Get details about a specific invitation.

Parameters
invitation_id (*str*) – invitation UUID

Return type
Invitation

listing(*workgroup_id*)

Returns pending invitations (including expired) for the specified workgroup.

Parameters
workgroup_id (*str*) – workgroup UUID

Return type
list

update(*invitation*)

Update a invitation owned by a invitation.

Parameters
invitation (*class*) – Invitation model object to update

Return type
Invitation

isogeo_pysdk.api.routes_keyword module

Isogeo API v1 - API Routes for Keywords entities

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.api.routes_keyword.**ApiKeyword**(*api_client=None*)

Bases: object

Routes as methods of Isogeo API used to manipulate keywords.

associate_workgroup(*workgroup*, *keyword*, *check_exists=1*)

Associate a keyword to a workgroup.

Parameters

- **workgroup** (Workgroup) – workgroup (resource) to edit
- **keyword** (Keyword) – object to associate

- **check_exists** (*bool*) – check if the keyword is already associated with the workgroup. Defaults to True.

Example

```
# retrieve a workgroup
wg = isogeo.workgroup.get(WORKGROUP_UUID)
# retrieve a keyword
kw = isogeo.keyword.get(KEYWORD_UUID)
# associate a keyword to a workgroup
isogeo.keyword.associate_workgroup(workgroup = wg, keyword = kw)
```

Return type
dict

create(*keyword*, *thesaurus_id='1616597fbc4348c8b11ef9d59cf594c8'*, *check_exists=0*)

Add a new keyword to the Isogeo thesaurus.

If a keyword with the same text already exists, the Isogeo API returns a 409 HTTP code. Then this method will try to get the closest matching keyword and return it.

Parameters

- **keyword** (Keyword) – Keyword model object to create
- **thesaurus_id** (*str*) – thesaurus UUID
- **check_exists** (*bool*) – check if a keyword with the same text already exists. Defaults to False.

Return type
Keyword

delete(*keyword*, *thesaurus_id='1616597fbc4348c8b11ef9d59cf594c8'*)

Delete a keyword from Isogeo database.

Parameters

- **keyword** (Keyword) – Keyword model object to create
- **thesaurus_id** (*str*) – thesaurus UUID

Return type
Keyword

dissociate_workgroup(*workgroup*, *keyword*, *check_exists=1*)

Dissociate a keyword from a workgroup.

Parameters

- **workgroup** (Workgroup) – workgroup (resource) to edit
- **keyword** (Keyword) – object to dissociate
- **check_exists** (*bool*) – check if the keyword is already dissociated with the workgroup. Defaults to True.

Example

```
# retrieve a workgroup
wg = isogeo.workgroup.get(WORKGROUP_UUID)
# retrieve a keyword
kw = isogeo.keyword.get(KEYWORD_UUID)
# dissociate a keyword to a workgroup
isogeo.keyword.dissociate_workgroup(workgroup = wg, keyword = kw)
```

Return type
dict

`get`(*keyword_id*, *include=('_abilities', 'count', 'thesaurus')*)
Get details about a specific keyword.

Parameters

- **keyword_id** (*str*) – keyword UUID
- **include** (*tuple*) – additionnal subresource to include in the response

Example

```
>>> # get a metadata with its tags (or keywords)
>>> md = isogeo.metadata.get(METADATA_UUID, include=("tags",))
>>> # list Isogeo keywords
>>> li_keywords_uuids = [
    tag[8:] for tag in self.metadata_source.tags
    if tag.startswith("keyword:isogeo:")
    ]
>>> # pick a random one
>>> random_keyword = sample(li_keywords_uuid, 1)[0]
>>> # get its details
>>> keyword = isogeo.keyword.get(random_keyword)
```

Return type
Keyword

`metadata`(*metadata_id=None*, *include=('_abilities', 'count', 'thesaurus')*)
List a metadata's keywords with complete information.

Parameters

- **metadata_id** (*str*) – metadata UUID
- **include** (*tuple*) – subresources that should be returned. Available values:
  - '_abilities'
  - 'count'
  - 'thesaurus'

Return type
list

`async search_keyword_asynchronous`(*total_results*, *max_workers=10*, ***kwargs*)
Meta async method used to request big searches (> 100 results), using asyncio. It's a private method launched by the main search method.

Parameters

- **total_results** (*int*) – total of results to retrieve
- **max_workers** (*int*) – maximum number of thread to use `python.concurrent.futures`

Return type
KeywordSearch

tagging(*metadata*, *keyword*, *check_exists=0*)

Associate a keyword to a metadata.

Parameters

- **metadata** (`Metadata`) – metadata (resource) to edit
- **keyword** (`Keyword`) – object to associate
- **check_exists** (*bool*) – check if the keyword is already tagged to the metadata. Defaults to False.

Example

```
# retrieve a metadata
md = isogeo.metadata.get(METADATA_UUID)
# retrieve a keyword
kw = isogeo.keyword.get(KEYWORD_UUID)
# associate a keyword to a metadata
isogeo.keyword.tagging(metadata = md, keyword = kw)
```

Return type
`dict`

thesaurus(*thesaurus_id='1616597fbc4348c8b11ef9d59cf594c8'*, *query=''*, *offset=0*, *order_by='text'*, *order_dir='desc'*, *page_size=100*, *specific_md=[]*, *specific_tag=[]*, *include=('_abilities', 'count')*, *whole_results=True*)

Search for keywords within a specific thesaurus or a specific group.

Parameters

- **thesaurus_id** (*str*) – thesaurus UUID
- **query** (*str*) – search terms, equivalent of **q** parameter in API.
- **offset** (*int*) – offset to start page size from a specific results index
- **order_by** (*str*) – sorting results. Available values:
  - 'count.isogeo': count of associated resources within Isogeo
  - 'text': alphabetical order [DEFAULT if relevance is null]
- **order_dir** (*str*) – sorting direction. Available values:
  - 'desc': descending [DEFAULT]
  - 'asc': ascending
- **page_size** (*int*) – limits the number of results. Default: 100.
- **specific_md** (*list*) – list of metadata UUIDs to filter on
- **specific_tag** (*list*) – list of tags UUIDs to filter on
- **include** (*tuple*) – subresources that should be returned. Available values:

- '_abilities'
- 'count'
- 'thesaurus'

- **whole_results** (*bool*) – option to return all results or only the page size. *False* by DEFAULT.

Return type
KeywordSearch

untagging(*metadata*, *keyword*)

Dissociate a keyword from a metadata.

Parameters

- **metadata** (Metadata) – metadata (resource) to edit
- **keyword** (Keyword) – object to associate

Return type
dict

workgroup(*workgroup_id=None*, *thesaurus_id=None*, *query=''*, *offset=0*, *order_by='text'*, *order_dir='desc'*, *page_size=100*, *specific_md=[]*, *specific_tag=[]*, *include=('_abilities', 'count', 'thesaurus')*, *whole_results=True*)

Search for keywords within a specific group's used thesauri.

Parameters

- **thesaurus_id** (*str*) – thesaurus UUID to filter on
- **query** (*str*) – search terms, equivalent of **q** parameter in API.
- **offset** (*int*) – offset to start page size from a specific results index
- **order_by** (*str*) – sorting results. Available values:
  - 'count.group': count of associated resources within the specified group
  - 'count.isogeo': count of associated resources within Isogeo
  - 'text': alphabetical order [DEFAULT]
- **order_dir** (*str*) – sorting direction. Available values:
  - 'desc': descending [DEFAULT]
  - 'asc': ascending
- **page_size** (*int*) – limits the number of results. Default: 100.
- **specific_md** (*list*) – list of metadata UUIDs to filter on
- **specific_tag** (*list*) – list of tags UUIDs to filter on
- **include** (*tuple*) – subresources that should be returned. Available values:
  - '_abilities'
  - 'count'
  - 'thesaurus'
- **whole_results** (*bool*) – option to return all results or only the page size. *False* by DEFAULT.

Return type
KeywordSearch

isogeo_pysdk.api.routes_license module

Isogeo API v1 - API Routes for Licenses (= CGUs, conditions) entities

See: http://help.isogeo.com/api/complete/index.html

`class isogeo_pysdk.api.routes_license.`**`ApiLicense`**`(`*api_client=None*`)`

Bases: `object`

Routes as methods of Isogeo API used to manipulate licenses (conditions).

`associate_metadata``(`*metadata*, *license*, *description*, *force=0*`)`

Associate a condition (license + specific description) to a metadata. When a license is associated to a metadata, it becomes a condition.

By default, if the specified license is already associated, the method won't duplicate the association. Use *force* option to overpass this behavior.

Parameters

- **`metadata`** (`Metadata`) – metadata object to update
- **`license`** (`License`) – license model object to associate
- **`description`** (*`str`*) – additional description to add to the association. Optional.
- **`force`** (*`bool`*) – force association even if the same license is already associated

Example

```
>>> # retrieve objects to be associated
>>> md = isogeo.metadata.get(
        metadata_id="6b5cc93626634d0e9b0d2c48eff96bc3",
        include=['conditions']
    )
>>> lic = isogeo.license.license("f6e0c665905a4feab1e9c1d6359a225f")
>>> # associate them
>>> isogeo.license.associate_metadata(
        metadata=md,
        license=lic,
        description="Specific description for this license when applied to this␣
↪metadata."
    )
```

Return type
`Response`

`create``(`*workgroup_id*, *check_exists=1*, *license={'_abilities': None, '_id': None, '_tag': None, 'content': None, 'count': None, 'link': None, 'name': None, 'owner': None}*`)`

Add a new license to a workgroup.

Parameters

- **`workgroup_id`** (*`str`*) – identifier of the owner workgroup
- **`check_exists`** (*`int`*) – check if a license already exists inot the workgroup:

- 0 = no check
- 1 = compare name [DEFAULT]

Parameters
`license` (*class*) – License model object to create

Return type
License

`delete`(*workgroup_id*, *license_id*)

Delete a license from Isogeo database.

Parameters

- `workgroup_id` (*str*) – identifier of the owner workgroup
- `license_id` (*str*) – identifier of the resource to delete

`exists`(*license_id*)

Check if the specified license exists and is available for the authenticated user.

Parameters
`license_id` (*str*) – identifier of the license to verify

Return type
`bool`

`get`(*license_id*)

Get details about a specific license.

Parameters
`license_id` (*str*) – license UUID

Return type
License

`listing`(*workgroup_id=None*, *include=('_abilities', 'count')*, *caching=1*)

Get workgroup licenses.

Parameters

- `workgroup_id` (*str*) – identifier of the owner workgroup
- `include` (*tuple*) – additionnal subresource to include in the response
- `caching` (*bool*) – option to cache the response

Return type
`list`

`update`(*license*, *caching=1*)

Update a license owned by a workgroup.

Parameters

- `license` (*class*) – License model object to update
- `caching` (*bool*) – option to cache the response

Return type
License

isogeo_pysdk.api.routes_limitation module

Isogeo API v1 - API Routes to manage metadata limitations.

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.api.routes_limitation.ApiLimitation(*api_client=None*)

Bases: `object`

Routes as methods of Isogeo API used to manipulate metadata limitations (CGUs).

create(*metadata*, *limitation*)

Add a new limitation to a metadata (= resource).

Parameters

- **metadata** (`Metadata`) – metadata (resource) to edit
- **limitation** (`Limitation`) – limitation object to create

Returns
409 if a limitation with the same name already exists.

Return type
Limitation or tuple

Example

```
# retrieve metadata
md = isogeo.metadata.get(METADATA_UUID)
# create the limitation locally
new_limitation = Limitation(
    type="legal",
    restriction="patent",
    description="Do not use for commercial purpose.",
    )
# add it to the metadata
isogeo.metadata.limitations.create(md, new_limitation)
```

delete(*limitation*, *metadata=None*)

Delete a limitation from a metadata.

Parameters

- **limitation** (`Limitation`) – Limitation model object to delete
- **metadata** (`Metadata`) – parent metadata (resource) containing the limitation

get(*metadata_id*, *limitation_id*)

Get details about a specific limitation.

Parameters

- **metadata_id** (*str*) – metadata UUID
- **limitation_id** (*str*) – limitation UUID

Example

```
>>> # get a metadata
>>> md = isogeo.metadata.get(METADATA_UUID)
>>> # list its limitations
>>> md_limitations = isogeo.metadata.limitations.listing(md)
>>> # get the first limitation
>>> limitation = isogeo.metadata.limitations.get(
    metadata_id=md._id,
    limitation_id=md_limitations[0].get("_id")
    )
```

Return type
`Limitation`

`listing`(*metadata*)
Get limitations of a metadata.

Parameters
metadata (`Metadata`) – metadata (resource)

Return type
`list`

`update`(*limitation*, *metadata=None*)
Update a limitation.

Parameters

- **limitation** (`Limitation`) – Limitation model object to update
- **metadata** (`Metadata`) – parent metadata (resource) containing the limitation

Return type
`Limitation`

isogeo_pysdk.api.routes_link module

Isogeo API v1 - API Routes to manage metadata links.

See: http://help.isogeo.com/api/complete/index.html

class `isogeo_pysdk.api.routes_link.ApiLink`(*api_client=None*)
Bases: `object`

Routes as methods of Isogeo API used to manipulate metadata links (CGUs).

`clean_kind_action_liability`(*link_actions*, *link_kind*)
Link available actions depend on link kind. Relationships between kinds and actions are described in the */link-kinds* route. This is a helper checking the liability between kind/actions/type and cleaning if needed. Useful before creating or updating a link.

Parameters

- **link_actions** (*list*) – link actions
- **link_kind** (*str*) – link kind

Return type
tuple

Example

```
# invalid action will be removed
print(isogeo.metadata.links.clean_kind_action_liability(
    link_actions=("download", "stream"),
    link_kind="url"
    )
)
>>> ('download',)
```

create(*metadata*, *link*, *clean_actions=True*)

Add a new link to a metadata (= resource).

Parameters

- **metadata** (Metadata) – metadata (resource) to edit
- **link** (Link) – link object to create
- **clean_actions** (*bool*) – if True, clean link actions depending on link kind

Returns
the created link or a tuple with the request's response error code

Return type
Link or tuple

Example

```
# retrieve metadata
md = isogeo.metadata.get(METADATA_UUID)
# create the link locally
new_link = Link(
    type="url",
    restriction="patent",
    description="Do not use for commercial purpose.",
    )
# add it to the metadata
isogeo.metadata.links.create(md, new_link)

# to create a link which is a pointer to another link, add the link attribute:
new_link = Link(
    actions=["other"],
    title="Associated link",
    kind="url",
    type="link",
    link=Link(_id=LINK_UUID)
    )
```

delete(*link*, *metadata=None*)

Delete a link from a metadata.

Parameters

- **link** (Link) – Link model object to delete
- **metadata** (Metadata) – parent metadata (resource) containing the link. Optional if the link contains the 'parent_resource' attribute.

Return type
Response

download_hosted(*link*, *encode_clean=1*)

Download hosted resource.

Parameters

- **link** (Link) – link object
- **encode_clean** (*bool*) – option to ensure a clean filename and avoid OS errors

Returns
tuple(stream, filename, human readable size)

Return type
tuple

Example:

```
# get links from a metadata
md_links = isogeo.metadata.links.listing(Metadata(_id=METADATA_UUID))
# filter on hosted links
hosted_links = [
    link for link in md_links
    if link.get("type") == "hosted"
    ]
# get the stream, the filename and the size (in human readable format)
dl_stream = isogeo.metadata.links.download_hosted(Link(**hosted_links[0]))
# download the file and store it locally
with open("./" + dl_stream[1], "wb") as fd:
    for block in dl_stream[0].iter_content(1024):
        fd.write(block)
```

get(*metadata_id*, *link_id*)

Get details about a specific link.

Parameters

- **metadata_id** (*str*) – metadata UUID
- **link_id** (*str*) – link UUID

Return type
Link

Example

```
# get a metadata
md = isogeo.metadata.get(METADATA_UUID)
# list its links
md_links = isogeo.metadata.links.listing(md)
# get the first link
link = isogeo.metadata.links.get(
    metadata_id=md._id,
    link_id=md_links[0].get("_id")
    )
```

kinds_actions(*caching=1*)

Get the relation between kinds and action for links.

Parameters
: **caching** (*bool*) – cache the response into the main API client instance. Defaults to True.

Returns
: list of dictionaries per link kinds

Return type
: list

Example

```
import pprint
pprint.pprint(isogeo.metadata.links.kinds_actions())

>>> [
        {
            'actions':
            [
                'download',
                'view',
                'other'
            ],
            'kind': 'url',
            'name': 'Lien'
        },
        {
            'actions': ['download', 'view', 'other'],
            'kind': 'wfs',
            'name': 'Service WFS'
        },
        {
            'actions': ['view', 'other'],
            'kind': 'wms',
            'name': 'Service WMS'
        },
        {
            'actions': ['view', 'other'],
            'kind': 'wmts',
            'name': 'Service WMTS'
        },
        {
            'actions': ['download', 'view', 'other'],
            'kind': 'esriFeatureService',
            'name': 'Service ESRI Feature'
        },
        {
            'actions': ['view', 'other'],
            'kind': 'esriMapService',
            'name': 'Service ESRI Map'
        },
        {
            'actions': ['view', 'other'],
            'kind': 'esriTileService',
            'name': 'Service ESRI Tile'
        },
```

(continues on next page)

(continued from previous page)

```
        {
            'actions': ['download', 'other'],
            'kind': 'data',
            'name': 'Donnée brute'
        }
    ]
```

listing(*metadata*)

Get links of a metadata.

Parameters
metadata (Metadata) – metadata (resource)

Return type
list

update(*link*, *metadata=None*)

Update a link.

Parameters

- **link** (Link) – Link model object to update
- **metadata** (Metadata) – parent metadata (resource) containing the link. Optional if the link contains the 'parent_resource' attribute.

Return type
Link

upload_hosted(*metadata*, *link*, *file_to_upload*)

Add a new link to a metadata uploading a file to hosted data. See: https://requests.readthedocs.io/en/latest/user/quickstart/#post-a-multipart-encoded-file

Parameters

- **metadata** (Metadata) – metadata (resource) to edit
- **link** (Link) – link object to create
- **file_to_upload** (*Path*) – file path to upload

Returns
the new Link if successed or the tuple with the request error code

Return type
Link or tuple

Example

```
from pathlib import Path

# define metadata
md = isogeo.metadata.get(METADATA_UUID)

# localize the file on the OS
my_file = Path("./upload/documentation.zip")

# create the link locally
lk = Link(
```

(continues on next page)

(continued from previous page)

```
    title=my_file.name
    )

# add it to the metadata
send = isogeo.metadata.links.upload_hosted(
    metadata=md,
    link=lk,
    file_to_upload=my_file.resolve()
    )
```

isogeo_pysdk.api.routes_metadata module

Isogeo API v1 - API Routes for Resources (= Metadata) entity

See: http://help.isogeo.com/api/complete/index.html#definition-resource

class isogeo_pysdk.api.routes_metadata.**ApiMetadata**(*api_client=None*)

Bases: `object`

Routes as methods of Isogeo API used to manipulate metadatas (resources).

catalogs(*metadata*)

Returns asssociated catalogs with a metadata. Just a shortcut.

Parameters

metadata (`Metadata`) – metadata object

Return type

list

create(*workgroup_id*, *metadata*, *return_basic_or_complete=0*)

Add a new metadata to a workgroup.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **metadata** (`Metadata`) – Metadata model object to create
- **return_basic_or_complete** (*int*) – creation of metada uses a bulk script. So, by default API does not return the complete object but the minimal info. This option allow to overrides the basic behavior. Options:
  - 0 = dry (only the _id, title and attributes passed for the creation) [DEFAULT]
  - 1 = basic without any include (requires an additionnal request)
  - 2 = complete with all include (requires an additionnal request)

Return type

Metadata

Example

```
# create a local metadata
my_metadata = Metadata(
    title="My awesome metadata",    # required
    type="vectorDataset",           # required
```

(continues on next page)

(continued from previous page)

```
    abstract="Here comes my **awesome** description with a piece of markdown."
↪# optional
)

# push it online
isogeo.metadata.create(
    workgroup_id=WORKGROUP_UUID,
    metadata=my_metadata
)
```

delete(*metadata_id*)

Delete a metadata from Isogeo database.

Parameters
metadata_id (*str*) – identifier of the resource to delete

Return type
Response

download_xml(*metadata*)

Download the metadata exported into XML ISO 19139.

Parameters
metadata (Metadata) – metadata object to export

Return type
Response

Example

```
# get the download stream
xml_stream = isogeo.metadata.download_xml(Metadata(_id=METADATA_UUID))
# write it to a file
with open("./{}.xml".format("metadata_exported_as_xml"), "wb") as fd:
    for block in xml_stream.iter_content(1024):
        fd.write(block)
```

exists(*resource_id*)

Check if the specified resource exists and is available for the authenticated user.

Parameters
resource_id (*str*) – identifier of the resource to verify

Return type
bool

get(*metadata_id*, *include=()*)

Get complete or partial metadata about a specific metadata (= resource).

Parameters

- **metadata_id** (*str*) – metadata UUID to get
- **include** (*tuple*) – subresources that should be included. Available values:
  - one or various from MetadataSubresources (Enum)
  - ”all” to get complete metadata with every subresource included

Return type
Metadata

keywords(*metadata*, *include=('_abilities', 'count', 'thesaurus')*)

Returns asssociated keywords with a metadata. Just a shortcut.

Parameters

- **metadata** (*Metadata*) – metadata object
- **include** (*tuple*) – subresources that should be returned. Available values:

- '_abilities'
- 'count'
- 'thesaurus'

Return type
list

update(*metadata*, *_http_method='PATCH'*)

Update a metadata, but **ONLY** the root attributes, not the subresources.

Certain attributes of the Metadata object to update are required:

- _id
- editionProfile
- type

See: https://github.com/isogeo/isogeo-api-py-minsdk/issues/116

Parameters

- **metadata** (*Metadata*) – metadata object to update
- **_http_method** (*str*) – HTTP method (verb) to use. Default to 'PATCH' but can be set to 'PUT' in certain cases (services).

Return type
Metadata

Returns
the updated metadata or the request error.

Example

```
# get a metadata
my_metadata = isogeo.metadata.get(metadata_id=METADATA_UUID)
# add an updated watermark in the abstract
my_metadata.abstract += '**Updated!**'
# push it online
isogeo.metadata.update(my_metadata)
```

isogeo_pysdk.api.routes_metadata_bulk module

Isogeo API v1 - API Route for bulk update on resources (= Metadata)

class isogeo_pysdk.api.routes_metadata_bulk.**ApiBulk**(*api_client=None*)

Bases: `object`

Routes as methods of Isogeo API used to mass edition of metadatas (resources).

Example

```
# retrieve objects to be associated
catalog_1 = isogeo.catalog.get(
    workgroup_id={WORKGROUP_UUID},
    catalog_id={CATALOG_UUID_1},
)

catalog_2 = isogeo.catalog.get(
    workgroup_id={WORKGROUP_UUID},
    catalog_id={CATALOG_UUID_2},
)

keyword = isogeo.keyword.get(keyword_id={KEYWORD_UUID},)

# along the script, prepare the bulk requests
isogeo.metadata.bulk.prepare(
    metadatas=(
        {METADATA_UUID_1},
        {METADATA_UUID_2},
    ),
    action="add",
    target="catalogs",
    models=(catalog_1, catalog_2),
)

isogeo.metadata.bulk.prepare(
    metadatas=(
        {METADATA_UUID_1},

    ),
    action="add",
    target="keywords",
    models=(keyword,),
)

# send the one-shot request
isogeo.metadata.bulk.send()
```

BULK_DATA = []

prepare(*metadatas*, *action*, *target*, *models*)

Prepare requests to be sent later in one shot.

Parameters

- **metadatas** (*tuple*) – tuple of metadatas UUIDs or Metadatas to be updated

- **action** (*str*) – type of action to perform on metadatas. See: *bulk_actions*.
- **target** (*str*) – kind of object to add/delete/update to the metadatas. See: *bulk_targets*.
- **models** (*tuple*) – tuple of objects to be associated with the metadatas.

Return type
BulkRequest

send()

Send prepared BULK_DATA to the *POST BULK resources/*.

Return type
List[*BulkReport*]

isogeo_pysdk.api.routes_search module

Isogeo API v1 - API Routes for Search

See: http://help.isogeo.com/api/complete/index.html#definition-resource

class isogeo_pysdk.api.routes_search.**ApiSearch**(*api_client=None*)

Bases: object

Routes as methods of Isogeo API used to manipulate metadatas (resources).

add_tags_shares(*search*)

Add shares list to the tags attributes in search.

Parameters
search (MetadataSearch) – search to add shares

search(*group=None*, *query=''*, *share=None*, *specific_md=()*, *include=()*, *bbox=None*, *poly=None*, *georel=None*, *order_by='_created'*, *order_dir='desc'*, *page_size=20*, *offset=0*, *augment=False*, *check=True*, *expected_total=None*, *tags_as_dicts=False*, *whole_results=False*)

Search within the resources shared to the application. It's the mainly used method to retrieve metadata.

Parameters

- **group** (*str*) – context to search. Pass a workgroup UUID to search within a group or pass None (default) to search in a global context.
- **query** (*str*) – search terms and semantic filters. Equivalent of **q** parameter in Isogeo API. It could be a simple string like *oil* or a tag like *keyword:isogeo:formations* or *keyword:inspire-theme:landcover*. The *AND* operator is applied when various tags are passed.
- **bbox** (*tuple*) – Bounding box to limit the search. Must be a 4 tuple of coordinates in WGS84 (EPSG 4326). Could be associated with *georel*.
- **poly** (*str*) – Geographic criteria for the search, in WKT format. Could be associated with *georel*.
- **georel** (*str*) – geometric operator to apply to the *bbox* or *poly* parameters. Available values:
  - 'contains',
  - 'disjoint',
  - 'equals',

  - 'intersects' - [APPLIED BY API if NOT SPECIFIED]
  - 'overlaps',
  - 'within'.
- **order_by** (*str*) – sorting results. Available values:
  - '_created': metadata creation date [DEFAULT if relevance is null]
  - '_modified': metadata last update
  - 'title': metadata title
  - 'created': data creation date (possibly None)
  - 'modified': data last update date
  - 'relevance': relevance score calculated by API [DEFAULT].
- **order_dir** (*str*) – sorting direction. Available values:
  - 'desc': descending
  - 'asc': ascending
- **page_size** (*int*) – limits the number of results. Useful to paginate results display. Default value: 100.
- **offset** (*int*) – offset to start page size from a specific results index
- **share** (*str*) – share UUID to filter on
- **specific_md** (*tuple*) – list of metadata UUIDs to filter on
- **include** (*tuple*) – subresources that should be returned. See: `enums.MetadataSubresources`.
- **whole_results** (*bool*) – option to return all results or only the page size. *False* by DEFAULT.
- **check** (*bool*) – option to check query parameters and avoid erros. *True* by DEFAULT.
- **augment** (*bool*) – option to improve API response by adding some tags on the fly (like shares_id)
- **expected_total** (*int*) – if different of None, value will be used to paginate. Can save a request.
- **tags_as_dicts** (*bool*) – option to store tags as key/values by filter.

Return type
MetadataSearch

Example

```
# get the search context (without results), useful to populate a search widget
search_context = isogeo.search(page_size=0, whole_results=0, augment=1)

# search the 10 first results in alphabetically order
search_10 = isogeo.search(
    page_size=10,
    include="all",
    order_by="title",
    order_dir="asc",
```

(continues on next page)

(continued from previous page)

```
    expected_total=search_context.total
)

# returns all results, filtering on vector-datasets
search_full = isogeo.search(
    query="type:vector-dataset",
    order_by="title",
    order_dir="desc",
    include="all",
    augment=1,
    whole_results=1
)
```

async search_metadata_asynchronous(*total_results*, *max_workers=10*, ***kwargs*)

Meta async method used to request big searches (> 100 results), using asyncio. It's a private method launched by the main search method.

Parameters

- **total_results** (*int*) – total of results to retrieve
- **max_workers** (*int*) – maximum number of thread to use `python.concurrent.futures`

Return type

MetadataSearch

isogeo_pysdk.api.routes_service module

Isogeo API v1 - API Routes for Metadata of Services (web geo services)

See: http://help.isogeo.com/api/complete/index.html#definition-resource

class isogeo_pysdk.api.routes_service.**ApiService**(*api_client=None*)

Bases: `object`

Routes as methods of Isogeo API used to manipulate metadatas of web geo services (services).

It's a set of helpers and shortcuts to make easier the sevrice management with the isogeo API.

create(*workgroup_id*, *service_url*, *service_type='guess'*, *service_format=None*, *service_title=None*, *check_exists=1*, *ignore_avaibility=0*, *http_verb='HEAD'*)

Add a new metadata of a geographic webservice to a workgroup.

It's just a helper to make it easy to create a metadata of a service with autofill for service layers.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **service_url** (*str*) – URL of the service
- **service_type** (*str*) – type of service. Must be one of: 'esri', 'esri_ogc', 'ogc', 'guess'
- **service_format** (*str*) – format of the web service. Must be one of the accepted codes in API (Non exhaustive list: 'efs', 'ems', 'wfs', 'wms', 'wmts'). If is None, so the it'll try to guess it from the URL.
- **service_title** (*str*) – title for the metadata service in case of analisis fail. OPTIONNAL.

- **check_exists** (*bool*) – check if a metadata with the same service base URL and format alerady exists. Defaults to True.
- **ignore_avaibility** (*bool*) – the service URL is tested to determine if it can be reached (HEAD then GET). This option allow to ignore the test result. Can be useful if the service is only reachable by certains URLs or domains like *.isogeo.com. Defaults to False.
- **http_verb** (*str*) – HTTP verb to use to check the if the service is available. Must be one of: GET, HEAD

Return type
Service

Raises

- **ValueError** – if workgroup_id is not a correct UUID | if http_verb or service_type is not a correct value
- ***AlreadyExistError*** – if a metadata service with the same base URL already exists in the workgroup

Example

```
# for an OGC WMS by GeoServer, passing type and format
isogeo.services.create(
    workgroup_id=WORKGROUP_UUID,
    service_type="ogc",
    service_format="wms",
    service_url="https://magosm.magellium.com/geoserver/ows?service=wms&
→version=1.3.0&request=GetCapabilities"
)
# for an OGC WFS by ArcGIS Server, passing only the service URL with query␣
→parameters
new_srv = isogeo.services.create(
    workgroup_id=WORKGROUP_UUID,
    service_url="https://ligeo.paysdelaloire.fr/server/services/Le_Mans/Le_Mans_
→service/MapServer/WFSServer?request=GetCapabilities&service=WFS",
)
# for an Esri FeatureServer
new_srv = isogeo.services.create(
    workgroup_id=WORKGROUP_UUID,
    service_url="https://api-carto.dijon.fr/arcgis/rest/services/SIGNALISATION/
→signalisation_MAJ/FeatureServer?f=pjson",
)
```

update(*service*, *check_only=0*)

Update a metadata of service while keeping the associations of the layers.

Parameters

- **metadata** (Metadata) – identifier of the resource to verify
- **check_only** (*bool*) – option to only get the diff

Return type
Metadata

isogeo_pysdk.api.routes_service_layers module

Isogeo API v1 - API Routes for ServiceLayers entities

See: http://help.isogeo.com/api/complete/index.html

`class isogeo_pysdk.api.routes_service_layers.`**`ApiServiceLayer`**`(`*api_client=None*`)`

Bases: `object`

Routes as methods of Isogeo API used to manipulate service_layers.

`associate_metadata``(`*service*, *layer*, *dataset*`)`

Associate a service layer with a dataset metadata.

If the specified layer is already associated, the response is 409.

Parameters

- **service** (`Metadata`) – metadata of the service which contains the layer
- **layer** (`ServiceLayer`) – layer model object to associate
- **dataset** (`Metadata`) – metadata of the dataset to associate with

Example

```
>>> # retrieve objects to be associated. First: the metadata of the service.
>>> service = isogeo.metadata.get(
        metadata_id=str,
    )
>>> # second: the layer of the service you want to associate
>>> layer = isogeo.metadata.layers.layer(
        metadata_id=service._id,
        layer_id=str,
    )
>>> # third: the dataset to be associated with the service layer
>>> dataset = isogeo.metadata.get(
        metadata_id=str,
    )
>>> # associate them
>>> isogeo.metadata.layers.associate_metadata(
        service=service,
        layer=layer,
        dataset=metadata
    )
```

Return type

`Response`

`create``(`*metadata*, *layer*`)`

Add a new layer to a metadata (= resource).

Parameters

- **metadata** (`Metadata`) – metadata (resource) to edit. Must be a service.
- **ServiceLayer** (`ServiceLayer`) – service_layer object to create

Return type

`ServiceLayer`

`delete`(*layer*, *metadata=None*)

Delete a service layer from Isogeo database.

Parameters

- **`layer`** (`ServiceLayer`) – ServiceLayer model object to delete
- **`metadata`** (`Metadata`) – parent metadata (resource) containing the service_layer

`dissociate_metadata`(*service*, *layer*, *dataset*)

Removes the association between a a service layer with a dataset metadata.

If the association doesn't exist, the response is 404.

Parameters

- **`service`** (`Metadata`) – metadata of the service which contains the layer
- **`layer`** (`ServiceLayer`) – layer model object to associate
- **`dataset`** (`Metadata`) – metadata of the dataset to associate with

Return type
`Response`

`layer`(*metadata_id*, *layer_id*)

Get details about a specific service_layer.

Parameters

- **`metadata_id`** (*`str`*) – metadata with layers
- **`layer_id`** (*`str`*) – service layer UUID

Return type
`ServiceLayer`

`listing`(*metadata*)

Get all service layers of a metadata.

Parameters
`metadata` (`Metadata`) – metadata (resource) to edit

Return type
`list`

`update`(*layer*, *metadata=None*)

Update a service layer.

Parameters

- **`layer`** (`ServiceLayer`) – ServiceLayer model object to update
- **`metadata`** (`Metadata`) – parent metadata (resource) containing the service_layer

Return type
`ServiceLayer`

isogeo_pysdk.api.routes_service_operations module

Isogeo API v1 - API Routes for ServiceOperations entities

See: http://help.isogeo.com/api/complete/index.html#tag-operation

`class isogeo_pysdk.api.routes_service_operations.`**`ApiServiceOperation`**`(`*api_client=None*`)`

Bases: `object`

Routes as methods of Isogeo API used to manipulate service_operations.

`create``(`*metadata*, *operation*`)`

Add a new operation to a metadata (= resource).

Parameters

- **`metadata`** (`Metadata`) – metadata (resource) to edit. Must be a service.
- **`ServiceOperation`** (`ServiceOperation`) – service_operation object to create

Return type

`ServiceOperation`

`listing``(`*metadata*`)`

Get all operations of a metadata service.

Parameters

`metadata` (`Metadata`) – metadata (resource) to edit. Must be type of service.

Return type

`list`

`operation``(`*metadata_id*, *operation_id*`)`

Get details about a specific service operation and expand the model with the parent service metadata '_id' reference.

Parameters

- **`metadata_id`** (*`str`*) – metadata with operations
- **`operation_id`** (*`str`*) – service operation UUID

Return type

`ServiceOperation`

isogeo_pysdk.api.routes_share module

Isogeo API v1 - API Routes for Shares entities

See: http://help.isogeo.com/api/complete/index.html

`class isogeo_pysdk.api.routes_share.`**`ApiShare`**`(`*api_client=None*`)`

Bases: `object`

Routes as methods of Isogeo API used to manipulate shares.

`associate_application``(`*share*, *application*`)`

Associate a share with an application.

Parameters

- **`share`** (`Share`) – share model object to update

- **application** (Application) – application object to associate

Return type
tuple

associate_catalog(*share*, *catalog*)

Associate a share with a catalog.

Parameters

- **share** (Share) – share model object to update
- **catalog** (Catalog) – object to associate

Return type
tuple

associate_group(*share*, *group*)

Associate a group with a share of type 'group'.

If the specified group is already associated, the response is still 204.

Parameters

- **share** (Share) – share model object to update
- **group** (Workgroup) – group object to associate

Return type
Response

create(*workgroup_id*, *share={'_created': None, '_creator': (None,), '_id': None, '_modified': None, 'applications': None, 'catalogs': None, 'groups': None, 'name': None, 'rights': None, 'type': None, 'urlToken': None}*, *check_exists=1*)

Add a new share to Isogeo.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **share** (Share) – Share model object to create
- **check_exists** (*int*) – check if a share already exists into the workgroup:

- 0 = no check
- 1 = compare name [DEFAULT]

Return type
Share

delete(*share_id*)

Delete a share from Isogeo database.

Parameters
share_id (*str*) – identifier of the resource to delete

Return type
Response

`dissociate_application`(*share*, *application*)

Removes the association between the specified share and the specified application.

Parameters

- **share** (Share) – share model object to update
- **application** (Application) – object to associate

Return type
tuple

`dissociate_catalog`(*share*, *catalog*)

Removes the association between the specified share and the specified catalog.

Parameters

- **share** (Share) – share model object to update
- **catalog** (Catalog) – object to associate

Return type
tuple

`dissociate_group`(*share*, *group*)

Removes the association between the specified share and the specified group.

If the specified group is associated, the association is removed, Response is 204. If not, the Response is 500.

Parameters

- **share** (Share) – share model object to update
- **group** (Workgroup) – object to associate

Return type
tuple

`exists`(*share_id*)

Check if the specified share exists and is available for the authenticated user.

Parameters
share_id (*str*) – identifier of the share to verify

Return type
bool

`get`(*share_id*, *include=('_abilities', 'groups')*)

Returns details about a specific share.

Parameters

- **share_id** (*str*) – share UUID
- **inlude** (*tuple*) – additionnal subresource to include in the response

Return type
Share

`listing`(*workgroup_id=None*, *caching=1*)

Get all shares which are accessible by the authenticated user OR shares for a workgroup.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup. If *None*, then list shares for the autenticated user
- **caching** (*bool*) – option to cache the response

Return type
`list`

refresh_token(*share*)

Refresh the URL token of a share, used by Cartotheque, CSW, OpenCatalog.

Parameters
share (`Share`) – Share model object to update

Return type
Share

reshare(*share*, *reshare=1*)

Enable/disable the reshare option for the given share.

Only available for shares of type 'group'.

Parameters

- **share** (`Share`) – Share model object to update
- **reshare** (*bool*) – set option to allow recipients groups

Return type
Share

update(*share*, *caching=1*)

Update a share owned by a workgroup.

Parameters

- **share** (`Share`) – Share model object to update
- **caching** (*bool*) – option to cache the response

Return type
Share

isogeo_pysdk.api.routes_specification module

Isogeo API v1 - API Routes for Specifications entities

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.api.routes_specification.**ApiSpecification**(*api_client=None*)

Bases: `object`

Routes as methods of Isogeo API used to manipulate specifications.

associate_metadata(*metadata*, *specification*, *conformity=0*)

Associate a specification (specification + conformity) to a metadata. When a specification is associated to a metadata, it becomes a ResourceConformity object.

If the specified specification is already associated, the API responses is still a 200.

Parameters

- **metadata** (`Metadata`) – metadata object to update

- **specification** (Specification) – specification model object to associate
- **conformity** (*bool*) – indicates whether the dataset is compliant

Example

```
>>> # retrieve objects to be associated
>>> md = isogeo.metadata.get(
        metadata_id=my_metadata_uuid,
        include=['specifications']
    )
>>> spec = isogeo.specification.get(my_specification_uuid)
>>> # associate them
>>> isogeo.specification.associate_metadata(
        metadata=md,
        specification=spec,
        conformity=1
    )
```

Return type
Response

create(*workgroup_id*, *check_exists=1*, *specification={'_abilities': None, '_id': None, '_tag': None, 'count': None, 'link': None, 'name': None, 'owner': None, 'published': None}*, *caching=1*)

Add a new specification to a workgroup.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **check_exists** (*int*) – check if a specification already exists inot the workgroup:

- 0 = no check
- 1 = compare name [DEFAULT]

Parameters

- **specification** (*class*) – Specification model object to create
- **caching** (*bool*) – option to cache the response

Return type
Specification

delete(*workgroup_id*, *specification_id*)

Delete a specification from Isogeo database.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **specification_id** (*str*) – identifier of the resource to delete

Return type
dict

dissociate_metadata(*metadata*, *specification_id*)

Removes the association between a metadata and a specification.

If the specified specification is not associated, the response is 404.

Parameters

- **metadata** (Metadata) – metadata object to update
- **specification_id** (Specification) – specification model object to associate

Return type
Response

exists(*specification_id*)

Check if the specified specification exists and is available for the authenticated user.

Parameters
specification_id (*str*) – identifier of the specification to verify

Return type
bool

get(*specification_id*)

Get a specification.

Parameters
specification_id (*str*) – specification UUID

Return type
Specification

listing(*workgroup_id=None*, *include=('_abilities', 'count')*, *caching=1*)

Get workgroup specifications.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **include** (*tuple*) – additional parts of model to include in response
- **caching** (*bool*) – option to cache the response

Return type
list

update(*specification*, *caching=1*)

Update a specification owned by a workgroup.

Parameters

- **specification** (*class*) – Specification model object to update
- **caching** (*bool*) – option to cache the response

Return type
Specification

isogeo_pysdk.api.routes_thesaurus module

Isogeo API v1 - API Routes for Thesaurus entities

See: http://help.isogeo.com/api/complete/index.html

`class isogeo_pysdk.api.routes_thesaurus.`**`ApiThesaurus`**`(`*api_client=None*`)`

Bases: `object`

Routes as methods of Isogeo API used to manipulate thesaurus.

`thesauri``(`*caching=1*`)`

Get all thesauri.

Return type
`list`

`thesaurus``(`*thesaurus_id='1616597fbc4348c8b11ef9d59cf594c8'*, *include=('_abilities',)*`)`

Get a thesaurus.

Parameters

- **`thesaurus_id`** (*`str`*) – thesaurus UUID
- **`include`** (*`list`*) – subresources that should be returned. Available values:
  - '_abilities'
  - 'count'

Return type
`Thesaurus`

isogeo_pysdk.api.routes_user module

Isogeo API v1 - API Routes for Users entities

See: http://help.isogeo.com/api/complete/index.html

`class isogeo_pysdk.api.routes_user.`**`ApiUser`**`(`*api_client=None*`)`

Bases: `object`

Routes as methods of Isogeo API used to manipulate users.

`create``(`*user=<class 'isogeo_pysdk.models.user.User'>*, *check_exists=1*`)`

Add a new user to Isogeo.

Parameters

- **`user`** (*`class`*) – User model object to create
- **`check_exists`** (*`bool`*) – check if a user already exists:

- 0 = no check
- 1 = compare email [DEFAULT]

Return type
`User`

delete(*user*)

Delete an user.

Parameters
user (*class*) – User model object to be deteled

Return type
User

get(*user_id*, *include='_abilities'*)

Get details about a specific user.

Parameters

- **user_id** (*str*) – user UUID
- **include** (*list*) – additionnal subresource to include in the response

Return type
User

listing()

Get registered users.

Example

```
>>> # get all registered users
>>> users = isogeo.user.listing()
>>> print(len(users))
925
>>> # filter on staff users (as list)
>>> staff = [user for user in users if user.get("staff")]
>>> print(len(staff))
10
>>> # filter on users with an email from isogeo(as list)
>>> users_isogeo = [user for user in users if "@isogeo" in user.get("contact").
↪get("email")]
>>> print(len(users_isogeo))
37
```

Return type
list

memberships(*user_id*)

Returns memberships for the specified user.

Parameters
user_id (*str*) – user UUID

Return type
dict

subscriptions(*user*, *subscription*, *subscribe*)

Subscribe or unsubscribe an user to/from one of the available subscriptions.

Parameters

- **user** (*class*) – User model object to be updated

- **subscription** (*str*) – subscription (newsletter) targetted. Must be one of: NewReleases | TipsAndTricks
- **subscribe** (*bool*) – subscribe (1) or unsubscribe (0)

Return type
User

Example

```
# retrieve the user
uzer = isogeo.user.get(user_id={user_uuid})

# unsubscribe the user from the newsletter 'TipsAndTricks'
isogeo.user.subscriptions(uzer, "TipsAndTricks", 0)

# subscribe the user to the newsletter 'NewReleases'
isogeo.user.subscriptions(uzer, "TipsAndTricks", 0)
```

update(*user*)

Update an user.

Parameters
user (*class*) – User model object to be updated

Return type
User

Example

```
# retrieve the user
uzer = isogeo.user.get(user_id={user_uuid})

# unsubscribe the user from a newsletter
uzer.mailchimp.get("subscriptions")[0]["isInterested"] = False

# update it online
isogeo.user.update(uzer)
```

isogeo_pysdk.api.routes_workgroup module

Isogeo API v1 - API Routes for Workgroups entities

See: http://help.isogeo.com/api/complete/index.html#tag-workgroup

class isogeo_pysdk.api.routes_workgroup.**ApiWorkgroup**(*api_client=None*)

Bases: object

Routes as methods of Isogeo API used to manipulate workgroups.

coordinate_systems(*workgroup_id*, *caching=1*)

Returns coordinate-systems for the specified workgroup. It's just an alias for the ApiCoordinateSystem.listing method.

Parameters

- **workgroup_id** (*str*) – workgroup UUID
- **caching** (*bool*) – option to cache the response

Return type
list

`create`(*workgroup*, *check_exists=1*)
Add a new workgroup to Isogeo.

Parameters

- **workgroup** (*class*) – Workgroup model object to create
- **check_exists** (*int*) – check if a workgroup already exists:
  - 0 = no check
  - 1 = compare name [DEFAULT]

Return type
Workgroup

`delete`(*workgroup_id*)
Delete a workgroup from Isogeo database.

Parameters
workgroup_id (*str*) – identifier of the workgroup

`exists`(*workgroup_id*)
Check if the specified workgroup exists and is available for the authenticated user.

Parameters
workgroup_id (*str*) – identifier of the workgroup to verify

Return type
bool

`get`(*workgroup_id*, *include=('_abilities', 'limits')*)
Get details about a specific workgroup.

Parameters

- **workgroup_id** (*str*) – workgroup UUID
- **include** (*tuple*) – additionnal subresource to include in the response

Return type
Workgroup

`invitations`(*workgroup_id*)
Returns active invitations (including expired) for the specified workgroup. Just a shortcut.

Parameters
workgroup_id (*str*) – workgroup UUID

Return type
dict

`invite`(*workgroup_id*, *invitation*)
Invite new user to a workgroup. Just a shortcut.

Parameters

- **workgroup_id** (*str*) – workgroup UUID
- **invitation** (Invitation) – Invitation object to send

Return type
`dict`

`limits`(*workgroup_id*)

Returns limits for the specified workgroup.

Parameters
`workgroup_id` (*`str`*) – workgroup UUID

Return type
`dict`

`listing`(*include=('_abilities', 'limits')*, *caching=1*)

Get workgroups.

Parameters

- **`include`** (*`list`*) – additionnal subresource to include in the response
- **`caching`** (*`bool`*) – option to cache the response

Return type
`list`

`memberships`(*workgroup_id*)

Returns memberships for the specified workgroup.

Parameters
`workgroup_id` (*`str`*) – workgroup UUID

Return type
`dict`

`statistics`(*workgroup_id*)

Returns statistics for the specified workgroup.

Parameters
`workgroup_id` (*`str`*) – workgroup UUID

Return type
`dict`

`statistics_by_tag`(*workgroup_id*, *tag*)

Returns statistics for the specified workgroup. See: http://help.isogeo.com/api/complete/index.html#operation–groups–gid–st
tag–tag–get

Be careful: if an invalid character is present into the response (e.g. contact.name = 'bureau GF-3A'), a ConnectionError / ReadTimeout will be raised.

Parameters

- **`workgroup_id`** (*`str`*) – workgroup UUID
- **`tag`** (*`str`*) – tag name. Must be one of: catalog, contact, coordinate-system, format, keyword:inspire-theme, keyword, owner

Return type
`dict`

`update`(*workgroup*, *caching=1*)

Update a workgroup owned by a workgroup.

Parameters

- **workgroup** (*class*) – Workgroup model object to update
- **caching** (*bool*) – option to cache the response

Return type
Workgroup

isogeo_pysdk.enums package

Submodules

isogeo_pysdk.enums.application_types module

Isogeo API v1 - Enums for Resource entity accepted kinds

See: http://help.isogeo.com/api/complete/index.html#definition-application

`class isogeo_pysdk.enums.application_types.ApplicationTypes(value)`

Bases: `Enum`

Closed list of accepted Application (metadata subresource) kinds in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for md_kind in ApplicationTypes:
>>>     print("{0:<30} {1:>20}".format(md_kind, md_kind.value))
Enum                                              Value
ApplicationTypes.group                                1
ApplicationTypes.user                                 2
```

```
>>> # check if a var is an accepted value
>>> print("group" in ApplicationTypes.__members__)
True
>>> print("User" in ApplicationTypes.__members__)  # case sensitive
False
>>> print("confidential" in ApplicationTypes.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

group = 1

user = 2

isogeo_pysdk.enums.bulk_actions module

Isogeo API v1 - Enums for Resource entity accepted kinds

See: http://help.isogeo.com/api/complete/index.html#definition-application

class isogeo_pysdk.enums.bulk_actions.**BulkActions**(*value*)

Bases: Enum

Closed list of accepted Application (metadata subresource) kinds in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for md_kind in BulkActions:
>>>     print("{0:<30} {1:>20}".format(md_kind, md_kind.value))
Enum                                          Value
BulkActions.add                                   1
BulkActions.delete                                2
BulkActions.update                                3
```

```
>>> # check if a var is an accepted value
>>> print("add" in BulkActions.__members__)
True
>>> print("Delete" in BulkActions.__members__)  # case sensitive
False
>>> print("truncate" in BulkActions.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

add = 1

delete = 2

update = 3

isogeo_pysdk.enums.bulk_ignore_reasons module

Isogeo API v1 - Enums for bulk ignore reasons

class isogeo_pysdk.enums.bulk_ignore_reasons.**BulkIgnoreReasons**(*value*)

Bases: Enum

Closed list of accepted Application (metadata subresource) kinds in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for md_kind in BulkIgnoreReasons:
>>>     print("{0:<30} {1:>20}".format(md_kind, md_kind.value))
Enum                                          Value
BulkIgnoreReasons.AlreadyPresent                  1
BulkIgnoreReasons.Forbidden                       2
```

(continues on next page)

(continued from previous page)

```
BulkIgnoreReasons.Invalid                              3
BulkIgnoreReasons.NotApplicable                        4
BulkIgnoreReasons.NotFound                             5
```

```
>>> # check if a var is an accepted value
>>> print("alreadyPresent" in BulkIgnoreReasons.__members__)
True
>>> print("NotValid" in BulkIgnoreReasons.__members__)  # case sensitive
False
>>> print("NotExists" in BulkIgnoreReasons.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

alreadyPresent = 1

forbidden = 2

invalid = 3

notApplicable = 4

notFound = 5

isogeo_pysdk.enums.bulk_targets module

Isogeo API v1 - Enums for Resource entity accepted kinds

See: http://help.isogeo.com/api/complete/index.html#definition-application

class isogeo_pysdk.enums.bulk_targets.**BulkTargets**(*value*)

Bases: `Enum`

Closed list of accepted Application (metadata subresource) kinds in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for md_kind in BulkTargets:
>>>     print("{0:<30} {1:>20}".format(md_kind, md_kind.value))
Enum                                              Value
BulkTargets.title                                     1
BulkTargets.abstract                                  2
BulkTargets.keywords                                  3
```

```
>>> # check if a var is an accepted value
>>> print("title" in BulkTargets.__members__)
True
>>> print("Delete" in BulkTargets.__members__)  # case sensitive
False
>>> print("truncate" in BulkTargets.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

abstract = 2

catalogs = 4

codePage = 12

collectionContext = 6

collectionMethod = 7

contacts = 5

distance = 10

keywords = 3

scale = 11

title = 1

validFrom = 8

validityComment = 9

isogeo_pysdk.enums.catalog_statistics_tags module

Isogeo API v1 - Enums for Catalog statistics entity accepted tags

See: http://help.isogeo.com/api/complete/index.html#operation–groups–gid–statistics-tag–tag–get

class isogeo_pysdk.enums.catalog_statistics_tags.**CatalogStatisticsTags**(*value*)

Bases: Enum

Closed list of accepted tags for workgroup statistics in Isogeo API (used by the dashboard).

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for tag in CatalogStatisticsTags:
>>>     print("{0:<30} {1:>20}".format(tag, tag.value))
Enum                                                    Value
CatalogStatisticsTags.catalog                                catalog
CatalogStatisticsTags.coordinateSystem             coordinate-system
CatalogStatisticsTags.format                                  format
CatalogStatisticsTags.inspireTheme             keyword:inspire-theme
CatalogStatisticsTags.owner                                    owner
```

```
>>> # check if a var is an accepted value
>>> print("catalog" in CatalogStatisticsTags.__members__)
True
>>> print("Catalog" in CatalogStatisticsTags.__members__)  # case␣
↪sensitive
False
>>> print("coordinate-system" in CatalogStatisticsTags.__members__)
```

(continues on next page)

(continued from previous page)

```
False
>>> print("coordinateSystem" in CatalogStatisticsTags.__members__)
True
```

See: https://docs.python.org/3/library/enum.html

contact = 'contact'

coordinateSystem = 'coordinate-system'

format = 'format'

classmethod has_value(*value*)

inspireTheme = 'keyword:inspire-theme'

keyword = 'keyword'

isogeo_pysdk.enums.contact_roles module

Isogeo API v1 - Enums for ResourceContact entity accepted roles

See: http://help.isogeo.com/api/complete/index.html#/definitions/resourceContact

class isogeo_pysdk.enums.contact_roles.**ContactRoles**(*value*)

Bases: Enum

Closed list of accepted Contact roles in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for role in ContactRoles:
>>>     print("{0:<30} {1:>20}".format(role, role.value))
Enum                                            Value
ContactRoles.author                            author
ContactRoles.pointOfContact              pointOfContact
...
```

```
>>> # check if a var is an accepted value
>>> print("author" in ContactRoles.__members__)
True
>>> print("Author" in ContactRoles.__members__)  # case sensitive
False
>>> print("follower" in ContactRoles.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

author = 'author'

custodian = 'custodian'

distributor = 'distributor'

originator = 'originator'

owner = 'owner'

pointOfContact = 'pointOfContact'

principalInvestigator = 'principalInvestigator'

processor = 'processor'

publisher = 'publisher'

resourceProvider = 'resourceProvider'

user = 'user'

isogeo_pysdk.enums.contact_types module

Isogeo API v1 - Enums for Contact entity accepted types

See: http://help.isogeo.com/api/complete/index.html#/definitions/resourceContact

class isogeo_pysdk.enums.contact_types.**ContactTypes**(*value*)

Bases: Enum

Closed list of accepted Contact types in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for i in ContactTypes:
>>>     print("{0:<30} {1:>20}".format(i, i.value))
Enum                                          Value
ContactTypes.custom                               1
ContactTypes.group                                2
ContactTypes.user                                 3
```

```
>>> # check if a var is an accepted value
>>> print("group" in ContactTypes.__members__)
True
>>> print("Custom" in ContactTypes.__members__)  # case sensitive
False
>>> print("global" in ContactTypes.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

custom = 1

group = 2

user = 3

isogeo_pysdk.enums.edition_profiles module

Isogeo API v1 - Enums for Resource entity accepted types

See: http://help.isogeo.com/api/complete/index.html#/definitions/resourceMetadata

class `isogeo_pysdk.enums.edition_profiles.`**`EditionProfiles`**`(`*value*`)`

Bases: `Enum`

Closed list of accepted edition profiles values in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for i in EditionProfiles:
>>>     print("{0:<30} {1:>20}".format(i, i.value))
Enum                                          Value
EditionProfiles.csw                               1
EditionProfiles.manual                            2
```

```
>>> # check if a var is an accepted value
>>> print("rasterDataset" in EditionProfiles.__members__)
True
>>> print("Service" in EditionProfiles.__members__)  # case sensitive
False
>>> print("dataset" in EditionProfiles.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

`csw = 1`

`manual = 2`

isogeo_pysdk.enums.event_kinds module

Isogeo API v1 - Enums for Resource entity accepted kinds

See: http://help.isogeo.com/api/complete/index.html#definition-resourceEvent

class `isogeo_pysdk.enums.event_kinds.`**`EventKinds`**`(`*value*`)`

Bases: `Enum`

Closed list of accepted Event (metadata subresource) kinds in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for md_kind in EventKinds:
>>>     print("{0:<30} {1:>20}".format(md_kind, md_kind.value))
Enum                                          Value
EventKinds.creation                               1
EventKinds.publication                            2
EventKinds.update                                 3
```

```
>>> # check if a var is an accepted value
>>> print("creation" in EventKinds.__members__)
True
>>> print("Update" in EventKinds.__members__)  # case sensitive
False
>>> print("modification" in EventKinds.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

creation = 1

publication = 2

update = 3

isogeo_pysdk.enums.keyword_casing module

Isogeo API v1 - Enums for Workgroup's keywords casing

See: http://help.isogeo.com/api/complete/index.html#definition-workgroup

class isogeo_pysdk.enums.keyword_casing.**KeywordCasing**(*value*)

Bases: Enum

Closed list of accepted Keyword casing in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for i in KeywordCasing:
>>>     print("{0:<30} {1:>20}".format(i, i.value))
Enum                                          Value
KeywordCasing.capitalized                         1
KeywordCasing.lowercase                           2
KeywordCasing.mixedcase                           3
KeywordCasing.uppercase                           4
```

```
>>> # check if a var is an accepted value
>>> print("capitalized" in KeywordCasing.__members__)
True
>>> print("Uppercase" in KeywordCasing.__members__)  # case sensitive
False
>>> print("initials" in KeywordCasing.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

capitalized = 1

lowercase = 2

mixedcase = 3

uppercase = 4

isogeo_pysdk.enums.limitation_restrictions module

Isogeo API v1 - Enums for Limitation restrictions entity accepted values.

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.enums.limitation_restrictions.**LimitationRestrictions**(*value*)

Bases: Enum

Closed list of accepted restrictions for limitations in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for tag in LimitationRestrictions:
>>>     print("{0:<30} {1:>20}".format(tag, tag.value))
Enum                                                        Value
LimitationRestrictions.copyright                            1
LimitationRestrictions.intellectualPropertyRights           2
LimitationRestrictions.license                              3
LimitationRestrictions.other                                4
LimitationRestrictions.patent                               5
LimitationRestrictions.patentPending                        6
LimitationRestrictions.trademark                            7
```

```
>>> # check if a var is an accepted value
>>> print("license" in LimitationRestrictions.__members__)
True
>>> print("License" in LimitationRestrictions.__members__)  # case␣
↪sensitive
False
>>> print("other" in LimitationRestrictions.__members__)
True
```

See: https://docs.python.org/3/library/enum.html

copyright = 1

intellectualPropertyRights = 2

license = 3

other = 4

patent = 5

patentPending = 6

trademark = 7

isogeo_pysdk.enums.limitation_types module

Isogeo API v1 - Enums for Limitation types entity accepted values.

See: http://help.isogeo.com/api/complete/index.html

class `isogeo_pysdk.enums.limitation_types.`**`LimitationTypes`**(*value*)

Bases: `Enum`

Closed list of accepted types for limitations in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for tag in LimitationTypes:
>>>     print("{0:<30} {1:>20}".format(tag, tag.value))
Enum                                          Value
LimitationTypes.legal                         1
LimitationTypes.security                      2
```

```
>>> # check if a var is an accepted value
>>> print("legal" in LimitationTypes.__members__)
True
>>> print("Legal" in LimitationTypes.__members__)  # case sensitive
False
>>> print("security" in LimitationTypes.__members__)
True
```

See: https://docs.python.org/3/library/enum.html

`legal = 1`

`security = 2`

isogeo_pysdk.enums.link_actions module

Isogeo API v1 - Enums for Links actions

See: http://help.isogeo.com/api/complete/index.html#definition-resourceLink

class `isogeo_pysdk.enums.link_actions.`**`LinkActions`**(*value*)

Bases: `Enum`

Closed list of accepted Link actions in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for i in LinkActions:
>>>     print("{0:<30} {1:>20}".format(i, i.value))
Enum                                 Value
LinkActions.download                     1
LinkActions.other                        2
LinkActions.view                         3
```

```
>>> # check if a var is an accepted value
>>> print("download" in LinkActions.__members__)
True
>>> print("Other" in LinkActions.__members__)  # case sensitive
False
>>> print("extract" in LinkActions.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

download = 1

other = 2

view = 3

isogeo_pysdk.enums.link_kinds module

Isogeo API v1 - Enums for Links kinds

See: http://help.isogeo.com/api/complete/index.html#definition-resourceLink

`class isogeo_pysdk.enums.link_kinds.LinkKinds(value)`

Bases: `Enum`

Closed list of accepted Link kinds in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for i in LinkKinds:
>>>     print("{0:<30} {1:>20}".format(i, i.value))
Enum                                          Value
LinkKinds.data                                    1
LinkKinds.esriFeatureService                      2
LinkKinds.esriMapService                          3
LinkKinds.esriTileService                         4
LinkKinds.url                                     5
LinkKinds.wfs                                     6
LinkKinds.wms                                     7
LinkKinds.wmts                                    8
```

```
>>> # check if a var is an accepted value
>>> print("data" in LinkKinds.__members__)
True
>>> print("EsriFeatureService" in LinkKinds.__members__)  # case␣
↪sensitive
False
>>> print("csw" in LinkKinds.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

data = 1

esriFeatureService = 2

esriMapService = 3

esriTileService = 4

url = 5

wfs = 6

wms = 7

wmts = 8

isogeo_pysdk.enums.link_types module

Isogeo API v1 - Enums for Links types

See: http://help.isogeo.com/api/complete/index.html#definition-resourceLink

class isogeo_pysdk.enums.link_types.**LinkTypes**(*value*)

Bases: Enum

Closed list of accepted Link types in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for i in LinkTypes:
>>>     print("{0:<30} {1:>20}".format(i, i.value))
Enum                                          Value
LinkTypes.hosted                                  1
LinkTypes.link                                    2
LinkTypes.url                                     3
```

```
>>> # check if a var is an accepted value
>>> print("hosted" in LinkTypes.__members__)
True
>>> print("Link" in LinkTypes.__members__)  # case sensitive
False
>>> print("external" in LinkTypes.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

hosted = 1

link = 2

url = 3

isogeo_pysdk.enums.metadata_subresources module

Isogeo API v1 - Enums for Metadata subresources

See: http://help.isogeo.com/api/complete/index.html#operation–resources–id–get

class isogeo_pysdk.enums.metadata_subresources.**MetadataSubresources**(*value*)

Bases: Enum

Closed list of accepted Metadata subresources that can be passed in *_include* queries paramater.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for i in MetadataSubresources:
>>>     print("{0:<30} {1:>20}".format(i, i.value))
Enum                                           Value
MetadataSubresources.tags                         1
MetadataSubresources.link                          2
MetadataSubresources.url                           3
```

```
>>> # check if a var is an accepted value
>>> print("tags" in MetadataSubresources.__members__)
True
>>> print("Links" in MetadataSubresources.__members__)  # case sensitive
False
>>> print("attributes" in MetadataSubresources.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

conditions = 'conditions'

contacts = 'contacts'

coordinateSystem = 'coordinate-system'

events = 'events'

featureAttributes = 'feature-attributes'

classmethod has_value(*value*)

keywords = 'keywords'

layers = 'layers'

limitations = 'limitations'

links = 'links'

operations = 'operations'

serviceLayers = 'serviceLayers'

specifications = 'specifications'

tags = 'tags'

isogeo_pysdk.enums.metadata_types module

Isogeo API v1 - Enums for Resource entity accepted types

See: http://help.isogeo.com/api/complete/index.html#/definitions/resourceMetadata

class isogeo_pysdk.enums.metadata_types.**MetadataTypes**(*value*)

Bases: Enum

Closed list of accepted Metadata (= Resource) types in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for md_type in MetadataTypes:
>>>     print("{0:<30} {1:>20}".format(md_type, md_type.value))
Enum                                           Value
MetadataTypes.noGeoDataset           no-geo-dataset
MetadataTypes.rasterDataset          raster-dataset
MetadataTypes.resource                     resource
MetadataTypes.service                       service
MetadataTypes.vectorDataset          vector-dataset
```

```
>>> # check if a var is an accepted value
>>> print("rasterDataset" in MetadataTypes.__members__)
True
>>> print("Service" in MetadataTypes.__members__)  # case sensitive
False
>>> print("dataset" in MetadataTypes.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

dataset = 'dataset'

classmethod has_value(*value*)

noGeoDataset = 'no-geo-dataset'

rasterDataset = 'raster-dataset'

resource = 'resource'

service = 'service'

vectorDataset = 'vector-dataset'

isogeo_pysdk.enums.search_filters_georelations module

Isogeo API v1 - Enums for Search geographic filter's geometric relationship

class isogeo_pysdk.enums.search_filters_georelations.**SearchGeoRelations**(*value*)

Bases: Enum

Closed list of accepted geometric relationship as search filters.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for i in SearchGeoRelations:
>>>     print("{0:<30} {1:>20}".format(i, i.value))
Enum                                            Value
SearchGeoRelations.contains                         1
SearchGeoRelations.disjoint                         2
SearchGeoRelations.equal                            3
SearchGeoRelations.intersects                       4
SearchGeoRelations.overlaps                         5
SearchGeoRelations.within                           6
```

```
>>> # check if a var is an accepted value
>>> print("contains" in SearchGeoRelations.__members__)
True
>>> print("Overlaps" in SearchGeoRelations.__members__)  # case␣
↪sensitive
False
>>> print("crosses" in SearchGeoRelations.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

contains = **1**

disjoint = **2**

equal = **3**

classmethod has_value(*value*)

intersects = **4**

overlaps = **5**

within = **6**

isogeo_pysdk.enums.session_status module

Isogeo API v1 - Enums for Session entity accepted status

See: http://help.isogeo.com/api/complete/index.html#definition-session

class isogeo_pysdk.enums.session_status.**SessionStatus**(*value*)

Bases: Enum

Closed list of accepted session (CSW) status values in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for i in SessionStatus:
>>>     print("{0:<30} {1:>20}".format(i, i.value))
Enum                                          Value
SessionStatus.canceled                            1
SessionStatus.failed                              2
SessionStatus.started                             3
SessionStatus.succeeded                           4
```

```
>>> # check if a var is an accepted value
>>> print("started" in SessionStatus.__members__)
True
>>> print("Failed" in SessionStatus.__members__)  # case sensitive
False
>>> print("aborted" in SessionStatus.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

canceled = 1

failed = 2

started = 3

succeeded = 4

isogeo_pysdk.enums.share_types module

Isogeo API v1 - Enums for Share entity accepted types.

See: http://help.isogeo.com/api/complete/index.html#definition-share

class isogeo_pysdk.enums.share_types.**ShareTypes**(*value*)

Bases: Enum

Closed list of accepted session (CSW) status values in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for i in ShareTypes:
>>>     print("{0:<30} {1:>20}".format(i, i.value))
Enum                                          Value
ShareTypes.canceled                             1
ShareTypes.failed                               2
ShareTypes.started                              3
ShareTypes.succeeded                            4
```

```
>>> # check if a var is an accepted value
>>> print("application" in ShareTypes.__members__)
True
>>> print("Group" in ShareTypes.__members__)  # case sensitive
False
>>> print("user" in ShareTypes.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

application = 1

group = 2

isogeo_pysdk.enums.user_roles module

Isogeo API v1 - Enums for ResourceContact entity accepted roles

See: http://help.isogeo.com/api/complete/index.html#definition-user

class isogeo_pysdk.enums.user_roles.**UserRoles**(*value*)

Bases: `Enum`

Closed list of accepted Contact roles in Isogeo API.

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for role in UserRoles:
>>>     print("{0:<30} {1:>20}".format(role, role.value))
Enum                                          Value
UserRoles.admin                          admin
UserRoles.writer          writer
...
```

```
>>> # check if a var is an accepted value
>>> print("admin" in UserRoles.__members__)
True
>>> print("Author" in UserRoles.__members__)  # case sensitive
False
>>> print("follower" in UserRoles.__members__)
False
```

See: https://docs.python.org/3/library/enum.html

admin = 'admin'

reader = 'reader'

writer = 'writer'

isogeo_pysdk.enums.workgroup_statistics_tags module

Isogeo API v1 - Enums for Workgroup statistics entity accepted tags

See: http://help.isogeo.com/api/complete/index.html#operation–groups–gid–statistics-tag–tag–get

class isogeo_pysdk.enums.workgroup_statistics_tags.**WorkgroupStatisticsTags**(*value*)

Bases: Enum

Closed list of accepted tags for workgroup statistics in Isogeo API (used by the dashboard).

Example

```
>>> # parse members and values
>>> print("{0:<30} {1:>20}".format("Enum", "Value"))
>>> for tag in WorkgroupStatisticsTags:
>>>     print("{0:<30} {1:>20}".format(tag, tag.value))
Enum                                                   Value
WorkgroupStatisticsTags.catalog                              catalog
WorkgroupStatisticsTags.coordinateSystem           coordinate-system
WorkgroupStatisticsTags.format                                format
WorkgroupStatisticsTags.inspireTheme           keyword:inspire-theme
WorkgroupStatisticsTags.owner                                  owner
```

```
>>> # check if a var is an accepted value
>>> print("catalog" in WorkgroupStatisticsTags.__members__)
True
>>> print("Catalog" in WorkgroupStatisticsTags.__members__)  # case␣
↪sensitive
False
>>> print("coordinate-system" in WorkgroupStatisticsTags.__members__)
False
>>> print("coordinateSystem" in WorkgroupStatisticsTags.__members__)
True
```

See: https://docs.python.org/3/library/enum.html

catalog = 'catalog'

contact = 'contact'

coordinateSystem = 'coordinate-system'

format = 'format'

classmethod has_value(*value*)

inspireTheme = 'keyword:inspire-theme'

```
keyword = 'keyword'

owner = 'owner'
```

isogeo_pysdk.models package

Submodules

isogeo_pysdk.models.application module

Isogeo API v1 - Model of Application entity

See: http://help.isogeo.com/api/complete/index.html#definition-application

class isogeo_pysdk.models.application.**Application**(*_abilities=None*, *_created=None*, *_id=None*, *_modified=None*, *canHaveManyGroups=None*, *client_id=None*, *client_secret=None*, *groups=None*, *kind=None*, *name=None*, *redirect_uris=None*, *scopes=None*, *staff=None*, *type=None*, *url=None*)

Bases: `object`

Applications are entities which can be used in shares.

Example

```
{
    "_abilities": [
        "application:delete",
        "application:manage",
        "application:update"
    ],
    "_created": "2018-02-13T16:53:37.4622+00:00",
    "_id": "2ad9ccd2c76a4fc3be9f8de4239701df",
    "_modified": "2018-02-13T16:53:43.085621+00:00",
    "canHaveManyGroups": true,
    "client_id": "plugin-arcmap-client-987a654z321e234r567t890y987u654i",
    "client_secret":
↪"LoremipsumdolorsitametconsecteturadipiscingelitDonecmaurismauris",
    "groups": [
        'groups': [{'_created': '2015-05-21T12:08:16.4295098+00:00',
        '_id': '32f7e95ec4e94ca3bc1afda960003882',
        '_modified': '2019-05-03T10:31:01.4796052+00:00',
        'canHaveManyGroups': 'groups:32f7e95ec4e94ca3bc1afda960003882',
        'areKeywordsRestricted': True,
        'canCreateLegacyServiceLinks': True,
        'canCreateMetadata': True,
        'contact': {'_deleted': False,
                    '_id': '2a3aefc4f80347f590afe58127f6cb0f',
                    'canHaveManyGroups':
↪'contact:group:2a3aefc4f80347f590afe58127f6cb0f',
                    'addressLine1': '26 rue du faubourg Saint-Antoine',
                    'addressLine2': '4 éme étage',
```

(continues on next page)

(continued from previous page)

```
                    'available': True,
                    'city': 'Paris',
                    'client_secretryCode': 'FR',
                    'email': 'dev@isogeo.com',
                    'fax': '33 (0)9 67 46 50 06',
                    'name': 'Isogeo Test',
                    'phone': '33 (0)9 67 46 50 06',
                    'type': 'group',
                    'zipCode': '75012'},
        'hasCswClient': True,
        'hasScanFme': True,
        'keywordsCasing': 'lowercase',
        'metadataLanguage': 'fr',
        'themeColor': '#4499A1'}
    ],
    "kind": "public",
    "name": "Plugin ArcMap - DEV",
    "scopes": [
        "resources:read"
    ],
    "staff": false,
    "type": "group",
    "url": "http://help.isogeo.com/arcmap/"
}
```

ATTR_CREA = {'canHaveManyGroups': <class 'bool'>, 'name': <class 'str'>, 'redirect_uris': <class 'list'>, 'scopes': <class 'list'>, 'staff': <class 'bool'>, 'type': <class 'str'>, 'url': <class 'str'>}

ATTR_MAP = {}

ATTR_TYPES = {'_abilities': <class 'list'>, '_created': <class 'str'>, '_id': <class 'str'>, '_modified': <class 'str'>, 'canHaveManyGroups': <class 'bool'>, 'client_id': <class 'str'>, 'client_secret': <class 'str'>, 'groups': <class 'list'>, 'kind': <class 'str'>, 'name': <class 'str'>, 'redirect_uris': <class 'list'>, 'scopes': <class 'list'>, 'staff': <class 'bool'>, 'type': <class 'str'>, 'url': <class 'str'>}

admin_url(*url_base='https://manage.isogeo.com'*)

Returns the administration URL (https://manage.isogeo.com) for this application.

Parameters

url_base (*str*) – base URL of admin site. Defaults to: https://manage.isogeo.com

Return type

str

property canHaveManyGroups: bool

Gets the option of this Application.

Returns

The option of this Application.

Return type

bool

property client_id: str

Gets the client_id of this Application.

Returns
The client_id of this Application.

Return type
str

property client_secret: str

Gets the client_secret of this Application.

Returns
The client_secret of this Application.

Return type
str

property groups

Gets the groups of this Application. # noqa: E501.

Returns
The groups of this Application. # noqa: E501

Return type
Workgroup

property kind: str

Gets the kind of this Application.

Returns
The kind of this Application.

Return type
str

property name: str

Gets the name of this Application.

Returns
The name of this Application.

Return type
str

property redirect_uris: list

Gets the redirect_uris of this Application.

Returns
The redirect_uris of this Application.

Return type
list

property scopes

Gets the scopes of this Application. # noqa: E501.

Returns
The scopes of this Application. # noqa: E501

Return type
Workgroup

property staff: bool

Gets the staff of this Application.

Returns
The staff of this Application.

Return type
bool

to_dict()

Returns the model properties as a dict.

Return type
dict

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()

Returns the string representation of the model.

Return type
str

property type

Gets the type of this Application. # noqa: E501.

Returns
The type of this Application. # noqa: E501

Return type
str

property url: str

Gets the url of this Application.

Returns
The url of this Application.

Return type
str

isogeo_pysdk.models.bulk_report module

Isogeo API v1 - Model of Metadata bulk report

See: https://github.com/isogeo/isogeo-api-py-minsdk/issues/133

class isogeo_pysdk.models.bulk_report.**BulkReport**(*ignored=None*, *request=None*)

Bases: object

Bulk report used to perform batch operation add/remove/update on Isogeo resources (= metadatas)

ATTR_TYPES = {'ignored': <class 'dict'>, 'request': <class 'isogeo_pysdk.models.bulk_request.BulkRequest'>}

property ignored: `dict`

Gets the ignored operations of this Bulk Request.

Returns
igno of this Bulk Request.

Return type
dict

property request: *BulkRequest*

Gets the created of this Bulk Request.

Returns
The created of this Bulk Request.

Return type
BulkRequest

to_dict()

Returns the request properties as a dict.

Return type
`dict`

to_str()

Returns the string representation of the request.

Return type
`str`

isogeo_pysdk.models.bulk_request module

Isogeo API v1 - Model of Metadata bulk request

See: https://github.com/isogeo/isogeo-api-py-minsdk/issues/133

class isogeo_pysdk.models.bulk_request.**BulkRequest**(*action=None*, *model=None*, *query=None*, *target=None*)

Bases: `object`

Bulk request used to perform batch operation add/remove/update on Isogeo resources (= metadatas)

ATTR_TYPES = {'action': <class 'object'>, 'model': <class 'int'>, 'query': <class 'dict'>, 'target': <class 'list'>}

property action: `str`

Gets the abilities of this Bulk Request.

Returns
The abilities of this Bulk Request.

Return type
str

property model: `list`

Gets the created of this Bulk Request.

Returns
The created of this Bulk Request.

Return type
list

property query: dict

Gets the modified of this Bulk Request.

Returns
The modified of this Bulk Request.

Return type
dict

property target: str

Gets the tag of this Bulk Request.

Returns
The tag of this Bulk Request.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
`dict`

to_str()

Returns the string representation of the model.

Return type
`str`

isogeo_pysdk.models.catalog module

Isogeo API v1 - Model of Catalog entity

See: http://help.isogeo.com/api/complete/index.html#definition-catalog

class isogeo_pysdk.models.catalog.**Catalog**(*_abilities=None*, *_created=None*, *_id=None*, *_modified=None*, *_tag=None*, *code=None*, *count=None*, *name=None*, *owner=None*, *scan=None*)

Bases: `object`

Catalogs are entities used to organize and shares metadata of a workgroup.

Example

```
{
    '$scan': boolean,
    '_abilities': array,
    '_created': string (datetime),
    '_id': string (uuid),
    '_modified': string (datetime),
    '_tag': string,
    'code': string,
    'count': integer,
    'name': string,
```

(continues on next page)

(continued from previous page)

```
    'owner': {
        '_created': string (datetime),
        '_id': string (uuid),
        '_modified': string (datetime),
        '_tag': string,
        'areKeywordsRestricted': boolean,
        'canCreateLegacyServiceLinks': boolean,
        'canCreateMetadata': boolean,
        'contact': {
            '_deleted': boolean,
            '_id': string (uuid),
            '_tag': string,
            'addressLine1': string,
            'addressLine2': string,
            'available': boolean,
            'city': string,
            'countryCode': string,
            'email': string (email),
            'fax': string,
            'name': string,
            'phone': string,
            'type': string,
            'zipCode': string
            },
        'hasCswClient': boolean,
        'hasScanFme': boolean,
        'keywordsCasing': string,
        'metadataLanguage': string
        }
}
```

ATTR_CREA = {'code': <class 'str'>, 'name': <class 'str'>, 'scan': <class 'bool'>}

ATTR_MAP = {'scan': '$scan'}

ATTR_TYPES = {'_abilities': <class 'list'>, '_created': <class 'str'>, '_id': <class 'str'>, '_modified': <class 'str'>, '_tag': <class 'str'>, 'code': <class 'str'>, 'count': <class 'int'>, 'name': <class 'str'>, 'owner': <class 'isogeo_pysdk.models.workgroup.Workgroup'>, 'scan': <class 'bool'>}

classmethod **clean_attributes**(*raw_object*)

Renames attributes wich are incompatible with Python (hyphens…).

See related issue: https://github.com/isogeo/isogeo-api-py-minsdk/issues/82

property **code**: str

Gets the code of this Catalog.

Returns

The code of this Catalog.

Return type

str

property count: str

Gets the count of this Catalog.

Returns
The count of this Catalog.

Return type
str

property name: str

Gets the name of this Catalog.

Returns
The name of this Catalog.

Return type
str

property owner

Gets the owner of this Catalog. # noqa: E501.

Returns
The owner of this Catalog. # noqa: E501

Return type
Workgroup

property scan: bool

Gets the scan of this Catalog.

Returns
The scan of this Catalog.

Return type
bool

to_dict()

Returns the model properties as a dict.

Return type
`dict`

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
`dict`

to_str()

Returns the string representation of the model.

Return type
`str`

isogeo_pysdk.models.condition module

Isogeo API v1 - Model of Condition entity

See: http://help.isogeo.com/api/complete/index.html#definition-resourceCondition

class isogeo_pysdk.models.condition.**Condition**(*_id=None*, *description=None*, *license=None*, *parent_resource=None*)

Bases: object

Conditions are entities defining general conditions of use (CGUs) of a data. It's mainly composed by a license and a description.

Parameters

- **_id** (*str*) – object UUID
- **description** (*str*) – description of the condition
- **license** (*dict*) – license object or dict linked to the condition
- **parent_resource** (*str*) – UUID of the metadata containing the condition

Example

```
{
    "_id": "string (uuid)",
    "description": "string",
    "license": "string",
}
```

ATTR_CREA = {'description': 'str', 'license': <class 'isogeo_pysdk.models.license.License'>}

ATTR_MAP = {}

ATTR_TYPES = {'_id': <class 'str'>, 'description': <class 'str'>, 'license': <class 'isogeo_pysdk.models.license.License'>, 'parent_resource': <class 'str'>}

property description: str

Gets the description of this Condition.

Returns
The description of this Condition.

Return type
str

property license: str

Gets the license of this Condition.

Returns
The license of this Condition.

Return type
str

property parent_resource

Gets the parent_resource of this Condition.

Returns
The parent_resource of this Condition.

Return type
UUID

to_dict()

Returns the model properties as a dict.

Return type
dict

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()

Returns the string representation of the model.

Return type
str

isogeo_pysdk.models.conformity module

Isogeo API v1 - Model of Conformity entity

See: http://help.isogeo.com/api/complete/index.html#definition-resourceConformity

class isogeo_pysdk.models.conformity.**Conformity**(*conformant=None*, *specification=None*, *parent_resource=None*)

Bases: object

Conformity is an entity defining if a data respects a specification. It's a quality indicator. It's mainly composed by a specification and a boolean.

Parameters

- **_id** (*str*) – object UUID
- **conformant** (*bool*) – conformity with the specification
- **specification** (*dict*) – specification object or dict linked to the conformity
- **parent_resource** (*str*) – UUID of the metadata containing the conformity

Example

```
{
    "conformant": "bool",
    "specification": "string",
}
```

ATTR_CREA = {'conformant': 'bool', 'specification': <class 'isogeo_pysdk.models.specification.Specification'>}

ATTR_MAP = {}

ATTR_TYPES = {'conformant': <class 'bool'>, 'parent_resource': <class 'str'>, 'specification': <class 'isogeo_pysdk.models.specification.Specification'>}

property conformant: bool

Gets the conformant status.

Returns
The conformant status

Return type
bool

property parent_resource

Gets the parent_resource of this Conformity.

Returns
The parent_resource of this Conformity.

Return type
UUID

property specification: *Specification*

Gets the specification of this Conformity.

Returns
The specification of this Conformity.

Return type
Specification

to_dict()

Returns the model properties as a dict.

Return type
`dict`

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
`dict`

to_str()

Returns the string representation of the model.

Return type
`str`

isogeo_pysdk.models.contact module

Isogeo API v1 - Model of Contact entity

See: http://help.isogeo.com/api/complete/index.html#definition-contact

class isogeo_pysdk.models.contact.**Contact**(*_abilities=None*, *_deleted=None*, *_id=None*, *_tag=None*, *addressLine1=None*, *addressLine2=None*, *addressLine3=None*, *available=None*, *city=None*, *count=None*, *countryCode=None*, *email=None*, *fax=None*, *hash=None*, *name=None*, *organization=None*, *owner=None*, *phone=None*, *type=None*, *zipCode=None*, *created=None*, *modified=None*)

Bases: `object`

Contacts are entities used into Isogeo adress book that can be associated to metadata.

ATTR_CREA = {'addressLine1': 'str', 'addressLine2': 'str', 'addressLine3': 'str', 'city': 'str', 'countryCode': 'str', 'email': 'str', 'fax': 'str', 'name': 'str', 'organization': 'str', 'phone': 'str', 'zipCode': 'str'}

ATTR_MAP = {'fax': 'faxNumber', 'organization': 'organizationName', 'phone': 'phoneNumber'}

ATTR_TYPES = {'_abilities': <class 'str'>, '_id': <class 'str'>, '_tag': <class 'str'>, 'addressLine1': <class 'str'>, 'addressLine2': <class 'str'>, 'addressLine3': <class 'str'>, 'city': <class 'str'>, 'count': <class 'int'>, 'countryCode': <class 'str'>, 'email': <class 'str'>, 'fax': <class 'str'>, 'hash': <class 'str'>, 'name': <class 'str'>, 'organization': <class 'str'>, 'owner': <class 'dict'>, 'phone': <class 'str'>, 'type': <class 'str'>, 'zipCode': <class 'str'>}

property addressLine1: `str`

Gets the id of this Contact.

Returns
The id of this Contact.

Return type
str

property addressLine2: `str`

Gets the id of this Contact.

Returns
The second address line of this Contact.

Return type
str

property addressLine3: `str`

Gets the third address line of this Contact.

Returns
The The third address line of this Contact.

Return type
str

property available: `bool`

Gets the availibility of this Contact.

Returns
The availibility of this Contact.

Return type
str

property city: `str`

Gets the city of this Contact.

Returns
The city of this Contact.

Return type
str

property count: int

Gets the id of this Contact.

Returns
The id of this Contact.

Return type
str

property countryCode: str

Gets the country code of this Contact.

Returns
The country code of this Contact.

Return type
str

property email: str

Gets the email of this Contact.

Returns
The email of this Contact.

Return type
str

property fax: str

Gets the fax of this Contact.

Returns
The fax of this Contact.

Return type
str

property hash: str

Gets the hash of this Contact.

Returns
The hash of this Contact.

Return type
str

property name: str

Gets the name of this Contact.

Returns
The name of this Contact.

Return type
str

property organization: str

Gets the organization of this Contact.

Returns
The organization of this Contact.

Return type
: str

property owner
: Gets the owner of this Specification.

  Returns
  : The owner of this Specification.

  Return type
  : *Workgroup*

property phone: str
: Gets the phone number of this Contact.

  Returns
  : The phone number of this Contact.

  Return type
  : str

to_dict()
: Returns the model properties as a dict.

  Return type
  : `dict`

to_dict_creation()
: Returns the model properties as a dict structured for creation purpose (POST)

  Return type
  : `dict`

to_str()
: Returns the string representation of the model.

  Return type
  : `str`

property type: str
: Gets the type of this Contact.

  Returns
  : The type of this Contact.

  Return type
  : str

property zipCode: str
: Gets the zip (postal) code of this Contact.

  Returns
  : The zip (postal) code of this Contact.

  Return type
  : str

isogeo_pysdk.models.coordinates_system module

Isogeo API v1 - Model of CoordinateSystem entity

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.models.coordinates_system.**CoordinateSystem**(*_tag=None*, *alias=None*, *code=None*, *name=None*)

Bases: `object`

CoordinateSystems.

Example

```
{
    '_tag': 'coordinate-system:31154',
    'code': 31154,
    'name': 'Zanderij / TM 54 NW'
}
```

ATTR_CREA = {}

ATTR_MAP = {}

ATTR_TYPES = {'_tag': <class 'str'>, 'alias': <class 'str'>, 'code': <class 'str'>, 'name': <class 'str'>}

property alias: str

Gets the custom alias of this CoordinateSystem in a workgroup.

Returns
The alias of this CoordinateSystem in a workgroup.

Return type
str

property code: str

Gets the EPSG code of this CoordinateSystem.

Returns
The EPSG code of this CoordinateSystem.

Return type
str

property name: str

Gets the name of this CoordinateSystem.

Returns
The name of this CoordinateSystem.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
`dict`

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()

Returns the string representation of the model.

Return type
str

isogeo_pysdk.models.datasource module

Isogeo API v1 - Model of Datasource entity

See: http://help.isogeo.com/api/complete/index.html#definition-datasource

class isogeo_pysdk.models.datasource.**Datasource**(*_created=None*, *_id=None*, *_modified=None*, *_tag=None*, *enabled=None*, *lastSession=None*, *location=None*, *name=None*, *resourceCount=None*, *sessions=None*)

Bases: object

Datasources are CSW client entry-points.

Example

```
{
    '_created': '2019-05-17T13:56:56.6162418+00:00',
    '_id': '2c891ce8692146c4901115a4232b13a2',
    '_modified': '2019-05-17T13:57:50.4434219+00:00',
    '_tag': 'data-source:2c891ce8692146c4901115a4232b13a2',
    'enabled': True,
    'lastSession': {
        '_created': '2019-05-17T13:58:06.5165889+00:00',
        '_id': 'ea99c37d809c4b1b9b4f257326ad1975',
        '_modified': '2019-05-17T13:58:28.5554966+00:00',
        'status': 'failed'
        },
    'location': 'http://ogc.geo-ide.developpement-durable.gouv.fr/csw/all-
↪harvestable',
    'name': 'TEST - CSW entrypoint (datasource)',
    'resourceCount': 0,
    'sessions': [
        {
            '_created': '2019-05-17T13:58:06.5165889+00:00',
            '_id': 'ea99c37d809c4b1b9b4f257326ad1975',
            '_modified': '2019-05-17T13:58:28.5554966+00:00',
            'status': 'failed'
        }]
}
```

ATTR_CREA = {'location': <class 'str'>, 'name': <class 'str'>}

ATTR_MAP = {}

ATTR_TYPES = {'_created': <class 'str'>, '_id': <class 'str'>, '_modified': <class 'str'>, '_tag': <class 'str'>, 'enabled': <class 'bool'>, 'lastSession': <class 'dict'>, 'location': <class 'str'>, 'name': <class 'str'>, 'resourceCount': <class 'bool'>, 'sessions': <class 'list'>}

property enabled: bool

Gets the enabled of this Datasource.

Returns
The enabled of this Datasource.

Return type
str

property lastSession: dict

Gets the lastSession of this Datasource.

Returns
The lastSession of this Datasource.

Return type
dict

property location: str

Gets the location (URL) of this Datasource.

Returns
The location (URL) of this Datasource.

Return type
str

property name: str

Gets the name of this Datasource.

Returns
The name of this Datasource.

Return type
str

property resourceCount: int

Gets the resourceCount of this Datasource.

Returns
The resourceCount of this Datasource.

Return type
Workgroup

property sessions: list

Gets the sessions of this Datasource.

Returns
The sessions of this Datasource.

Return type
Workgroup

to_dict()

Returns the model properties as a dict.

Return type
dict

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()

Returns the string representation of the model.

Return type
str

isogeo_pysdk.models.directive module

Isogeo API v1 - Model of Directive entity

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.models.directive.**Directive**(*_id=None*, *description=None*, *name=None*)

Bases: object

Directives are entities included as subresource of limitations into metadata CGUs.

Example

```
{
    "_id": string (uuid),
    "name": string,
    "description": string
}
```

ATTR_MAP = {}

ATTR_TYPES = {'_id': <class 'str'>, 'description': <class 'str'>, 'name': <class 'str'>}

property description: str

Gets the description of this Directive.

Returns
The description of this Directive.

Return type
str

property name: str

Gets the name of this Directive.

Returns
The name of this Directive.

Return type
str

`to_dict()`

Returns the model properties as a dict.

Return type
dict

`to_str()`

Returns the string representation of the model.

Return type
str

isogeo_pysdk.models.event module

Isogeo API v1 - Model of Event entity

See: http://help.isogeo.com/api/complete/index.html#definition-event

class `isogeo_pysdk.models.event.Event`(*_id=None*, *date=None*, *description=None*, *kind=None*, *parent_resource=None*, *waitForSync=1*)

Bases: `object`

Events are entities included as subresource into metadata for data history description.

Example

```
{
    '_id': string (uuid),
    'date': string (datetime),
    'description': string,
    'kind': string
}
```

`ATTR_CREA = {'date': <class 'str'>, 'description': <class 'str'>, 'kind': <class 'str'>, 'waitForSync': <class 'bool'>}`

`ATTR_MAP = {}`

`ATTR_TYPES = {'_id': <class 'str'>, 'date': <class 'str'>, 'description': <class 'str'>, 'kind': <class 'str'>, 'parent_resource': <class 'str'>}`

`property date: str`

Gets the date of this Event.

Returns
The date of this Event.

Return type
str

`property description: str`

Gets the description of this Event.

Returns
The description of this Event.

Return type
str

property kind: str

Gets the kind of this Event.

Returns
The kind of this Event.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
dict

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()

Returns the string representation of the model.

Return type
str

isogeo_pysdk.models.feature_attributes module

Isogeo API v1 - Model of FeatureAttributes entity

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.models.feature_attributes.**FeatureAttribute**(*_id=None*, *alias=None*, *comment=None*, *dataType=None*, *description=None*, *isAutoGenerated=None*, *isNullable=None*, *isReadOnly=None*, *hasElevation=None*, *hasMeasure=None*, *language=None*, *length=None*, *name=None*, *precision=None*, *propertyType=None*, *scale=None*, *spatialContext=None*, *parent_resource=None*)

Bases: object

FeatureAttributes are entities included as subresource into metadata.

Parameters

- **_id** (*str*) – UUID, defaults to None
- **alias** (*str*) – alias of the feature attribute, defaults to None
- **comment** (*str*) – comment of the feature attribute, defaults to None
- **dataType** (*str*) – kind of field (varchar, integer32…), defaults to None

- **description** (*str*) – description of the feature attribute, defaults to None
- **language** (*str*) – language of the description, defaults to None
- **length** (*int*) – length of the values accepted in the attribute, defaults to None
- **name** (*str*) – attribute name, defaults to None
- **precision** (*int*) – value precision, defaults to None
- **scale** (*int*) – scale of display, defaults to None

Example

```
{
    "_id": string (uuid),
    "alias": string,
    "comment": string,
    "dataType": string,
    "description": string,
    "language": string,
    "length": int,
    "name": string,
    "precision": int,
    "scale": int,
}
```

ATTR_CREA = {'alias': <class 'str'>, 'comment': <class 'str'>, 'dataType': <class 'str'>, 'description': <class 'str'>, 'hasElevation': <class 'bool'>, 'hasMeasure': <class 'bool'>, 'isAutoGenerated': <class 'bool'>, 'isNullable': <class 'bool'>, 'isReadOnly': <class 'bool'>, 'language': <class 'str'>, 'length': <class 'int'>, 'name': <class 'str'>, 'precision': <class 'int'>, 'propertyType': <class 'str'>, 'scale': <class 'int'>, 'spatialContext': <class 'str'>}

ATTR_MAP = {}

ATTR_TYPES = {'_id': <class 'str'>, 'alias': <class 'str'>, 'comment': <class 'str'>, 'dataType': <class 'str'>, 'description': <class 'str'>, 'hasElevation': <class 'bool'>, 'hasMeasure': <class 'bool'>, 'isAutoGenerated': <class 'bool'>, 'isNullable': <class 'bool'>, 'isReadOnly': <class 'bool'>, 'language': <class 'str'>, 'length': <class 'int'>, 'name': <class 'str'>, 'parent_resource': <class 'str'>, 'precision': <class 'int'>, 'propertyType': <class 'str'>, 'scale': <class 'int'>, 'spatialContext': <class 'str'>}

property alias: str

Gets the alias of this FeatureAttribute.

Returns
The alias of this FeatureAttribute.

Return type
str

property comment: str

Gets the comment of this FeatureAttribute.

Returns
The comment of this FeatureAttribute.

Return type
str

property dataType: str

Gets the dataType of this FeatureAttribute.

Returns
The dataType of this FeatureAttribute.

Return type
str

property description: str

Gets the description of this FeatureAttribute.

Returns
The description of this FeatureAttribute.

Return type
str

property hasElevation: bool

Gets the hasElevation of this FeatureAttribute.

Returns
The hasElevation of this FeatureAttribute.

Return type
bool

property hasMeasure: bool

Gets the hasMeasure of this FeatureAttribute.

Returns
The hasMeasure of this FeatureAttribute.

Return type
bool

property isAutoGenerated: bool

Gets the isAutoGenerated of this FeatureAttribute.

Returns
The isAutoGenerated of this FeatureAttribute.

Return type
bool

property isNullable: bool

Gets the isNullable of this FeatureAttribute.

Returns
The isNullable of this FeatureAttribute.

Return type
bool

property isReadOnly: bool

Gets the isReadOnly of this FeatureAttribute.

Returns
The isReadOnly of this FeatureAttribute.

Return type
bool

property language: str

Gets the language of this FeatureAttribute.

Returns
The language of this FeatureAttribute.

Return type
str

property length: int

Gets the length of this FeatureAttribute.

Returns
The length of this FeatureAttribute.

Return type
int

property name: str

Gets the name of this FeatureAttribute.

Returns
The name of this FeatureAttribute.

Return type
str

property precision: int

Gets the precision of this FeatureAttribute.

Returns
The precision of this FeatureAttribute.

Return type
int

property propertyType: str

Gets the propertyType of this FeatureAttribute.

Returns
The propertyType of this FeatureAttribute.

Return type
str

property scale: int

Gets the scale of this FeatureAttribute.

Returns
The scale of this FeatureAttribute.

Return type
int

property spatialContext: str

Gets the spatialContext of this FeatureAttribute.

Returns
The spatialContext of this FeatureAttribute.

Return type
str

to_dict()
Returns the model properties as a dict.

Return type
dict

to_dict_creation()
Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()
Returns the string representation of the model.

Return type
str

isogeo_pysdk.models.format module

Isogeo API v1 - Model of Format entity

See: http://help.isogeo.com/api/complete/index.html#definition-format

class isogeo_pysdk.models.format.**Format**(*_id=None*, *_tag=None*, *aliases=None*, *code=None*, *name=None*, *type=None*, *versions=None*)

Bases: object

Formats are entities included as subresource into metadata for data history code.

Example

```
{
    "_id": string (uuid),
    "_tag": "format:dgn",
    "aliases": [
        "dgnv7",
        "dgnv8",
        "igds"
    ],
    "code": "dgn",
    "name": "DGN",
    "type": "dataset",
    "versions": [
        "v8",
        "V7",
        null
    ]
}
```

ATTR_CREA = {'aliases': <class 'list'>, 'code': <class 'str'>, 'name': <class 'str'>, 'type': <class 'str'>, 'versions': <class 'list'>}

ATTR_MAP = {}

ATTR_TYPES = {'_id': <class 'str'>, '_tag': <class 'str'>, 'aliases': <class 'list'>, 'code': <class 'str'>, 'name': <class 'str'>, 'type': <class 'str'>, 'versions': <class 'list'>}

property aliases: list

Gets the aliases of this Format.

Returns
The aliases of this Format.

Return type
list

property code: str

Gets the code of this Format.

Returns
The code of this Format.

Return type
str

property name: str

Gets the name of this Format.

Returns
The name of this Format.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
dict

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()

Returns the string representation of the model.

Return type
str

property type: str

Gets the type of this Format.

Returns
The type of this Format.

Return type
str

property versions: list

Gets the versions of this Format.

Returns
The versions of this Format.

Return type
list

isogeo_pysdk.models.invitation module

Isogeo API v1 - Model of Invitation entity

See: http://help.isogeo.com/api/complete/index.html#definition-invitation

class isogeo_pysdk.models.invitation.Invitation(*_created=None*, *_id=None*, *_modified=None*, *email=None*, *expiresIn=None*, *group=None*, *role=None*)

Bases: object

Invitations are CSW client entry-points.

Example

```
{
    "_id": "6c7c9e0c63a943f79ba1e00766d0082d",
    "_created": "2019-07-25T09:23:37.0975771+00:00",
    "_modified": "2019-07-25T09:23:37.0975771+00:00",
    "role": "admin",
    "email": "prenom.nom@organisation.code",
    "expiresIn": 657364,
    "group": {
        "_id": "string (uuid)",
        "_tag": "owner:string (uuid)",
        "_created": "2019-05-07T15:11:08.5202923+00:00",
        "_modified": "2019-07-25T09:13:29.7858081+00:00",
        "contact": {
            "_id": "string (uuid)",
            "_tag": "contact:group:string (uuid)",
            "_deleted": false,
            "type": "group",
            "group": "Isogeo TEST",
            "available": false
        },
        "canCreateMetadata": true,
        "canCreateLegacyServiceLinks": false,
        "areKeywordsRestricted": false,
        "hasCswClient": false,
        "hasScanFme": false,
        "keywordsCasing": "lowercase"
    }
```

ATTR_CREA = {'email': <class 'str'>, 'group': <class 'str'>, 'role': <class 'str'>}

`ATTR_MAP = {}`

`ATTR_TYPES = {'_created': <class 'str'>, '_id': <class 'str'>, '_modified': <class 'str'>, 'email': <class 'str'>, 'expiresIn': <class 'int'>, 'group': <class 'isogeo_pysdk.models.workgroup.Workgroup'>, 'role': <class 'str'>}`

property email: str

Gets the email of this Invitation.

Returns
The email of this Invitation.

Return type
str

property expiresIn: int

Gets the expiresIn of this Invitation.

Returns
The expiresIn of this Invitation.

Return type
int

property group: *Workgroup*

Gets the group of this Invitation.

Returns
The group of this Invitation.

Return type
Workgroup

property role: bool

Gets the role of this Invitation.

Returns
The role of this Invitation.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
`dict`

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
`dict`

to_str()

Returns the string representation of the model.

Return type
`str`

isogeo_pysdk.models.keyword module

Isogeo API v1 - Model of Keyword entity

See: http://help.isogeo.com/api/complete/index.html#definition-keyword

class isogeo_pysdk.models.keyword.**Keyword**(*_abilities=None*, *_created=None*, *_id=None*, *_modified=None*, *_tag=None*, *code=None*, *count=None*, *description=None*, *thesaurus=None*, *text=None*)

Bases: object

Keywords are entities used to organize and shares metadata of a workgroup.

Example

```
{
    '_abilities': [
        'keyword:delete',
        'keyword:restrict'
        ],
    '_created': None,
    '_id': 'ac56a9fbe6f348a79ec9899ebce2d6da',
    '_modified': None,
    '_tag': 'keyword:isogeo:tests-unitaires',
    'code': 'tests-unitaires',
    'count': {
        'isogeo': 0
        },
    'description': None,
    'text': 'tests unitaires',
    'thesaurus': {
        '_id': '1616597fbc4348c8b11ef9d59cf594c8',
        'code': 'isogeo'
        }
}
```

ATTR_CREA = {'text': <class 'str'>}

ATTR_MAP = {}

ATTR_TYPES = {'_abilities': <class 'list'>, '_created': <class 'str'>, '_id': <class 'str'>, '_modified': <class 'str'>, '_tag': <class 'str'>, 'code': <class 'str'>, 'count': <class 'dict'>, 'description': <class 'str'>, 'text': <class 'str'>, 'thesaurus': <class 'isogeo_pysdk.models.thesaurus.Thesaurus'>}

property code: str

Gets the code of this Keyword.

Returns
The code of this Keyword.

Return type
str

property count: dict

Gets the count of this Keyword.

Returns
The count of this Keyword.

Return type
dict

property description: str

Gets the description of this Keyword.

Returns
The description of this Keyword.

Return type
str

property text: bool

Gets the text of this Keyword.

Returns
The text of this Keyword.

Return type
bool

property thesaurus

Gets the thesaurus of this Keyword. # noqa: E501.

Returns
The thesaurus of this Keyword. # noqa: E501

Return type
Thesaurus

to_dict()

Returns the model properties as a dict.

Return type
`dict`

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
`dict`

to_str()

Returns the string representation of the model.

Return type
`str`

isogeo_pysdk.models.keyword_search module

Isogeo API v1 - Model of Keyword search entity

See: http://help.isogeo.com/api/complete/index.html#definition-search

class isogeo_pysdk.models.keyword_search.**KeywordSearch**(*limit=None*, *offset=None*, *results=None*, *total=None*)

Bases: `object`

Keyword searchs are entities used to organize and shares metadata of a workgroup.

ATTR_TYPES = {'limit': <class 'int'>, 'offset': <class 'int'>, 'results': <class 'list'>, 'total': <class 'int'>}

property limit: int

Gets the created of this Keyword search.

Returns
The created of this Keyword search.

Return type
str

property offset: int

Gets the offset of this Keyword search.

Returns
The offset of this Keyword search.

Return type
int

property results: str

Gets the tag of this Keyword search.

Returns
The tag of this Keyword search.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
`dict`

to_str()

Returns the string representation of the model.

Return type
`str`

property total: str

Gets the total of this Keyword search.

Returns
The total of this Keyword search.

Return type
str

isogeo_pysdk.models.license module

Isogeo API v1 - Model of License entity

See: http://help.isogeo.com/api/complete/index.html#definition-license

`class isogeo_pysdk.models.license.License(_abilities=None, _id=None, _tag=None, count=None, content=None, link=None, name=None, owner=None)`

Bases: `object`

Licenses are entities included as subresource into metadata.

Example

```
{
    "_id": "string (uuid)",
    "content": "string",
    "count": "integer (int32)",
    "link": "string",
    "name": "string"
}
```

Attributes:
ATTR_TYPES (dict): basic structure of license attributes. {"attribute name": "attribute type"}. ATTR_CREA (dict): only attributes used to POST requests. {"attribute name": "attribute type"}

`ATTR_CREA = {'content': 'str', 'link': 'str', 'name': 'str'}`

`ATTR_MAP = {}`

`ATTR_TYPES = {'_abilities': <class 'str'>, '_id': <class 'str'>, '_tag': <class 'str'>, 'content': <class 'str'>, 'count': <class 'int'>, 'link': <class 'str'>, 'name': <class 'str'>, 'owner': <class 'dict'>}`

`property content: str`

Gets the content of this License.

Returns
The content of this License.

Return type
str

`property count: int`

Gets the count of this License.

Returns
The count of this License.

Return type
int

property link: str

Gets the link (URL) of this License.

Returns
The link (URL) of this License.

Return type
str

property name: str

Gets the name of this License.

Returns
The name of this License.

Return type
str

property owner

Gets the owner of this License.

Returns
The owner of this License.

Return type
Workgroup

to_dict()

Returns the model properties as a dict.

Return type
`dict`

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
`dict`

to_str()

Returns the string representation of the model.

Return type
`str`

isogeo_pysdk.models.limitation module

Isogeo API v1 - Model of Limitation entity

See: http://help.isogeo.com/api/complete/index.html

class isogeo_pysdk.models.limitation.**Limitation**(*_id=None*, *description=None*, *directive=None*, *restriction=None*, *type=None*, *parent_resource=None*)

Bases: `object`

Limitations are entities included as subresource into metadata which can contain a Directive.

Example

```
{
    "_id": "string (uuid)",
    "description": "string",
    "directive": "dict",
    "restriction": "string",
    "type": "string"
}
```

ATTR_CREA = {'description': 'str', 'directive': <class 'isogeo_pysdk.models.directive.Directive'>, 'restriction': 'str', 'type': 'str'}

ATTR_MAP = {}

ATTR_TYPES = {'_id': <class 'str'>, 'description': <class 'str'>, 'directive': <class 'int'>, 'parent_resource': <class 'str'>, 'restriction': <class 'str'>, 'type': <class 'str'>}

property description: str

Gets the description of this Limitation.

Returns
The description of this Limitation.

Return type
str

property directive: *Directive*

Gets the directive of this Limitation.

Returns
The directive of this Limitation.

Return type
Directive

property restriction: str

Gets the restriction of this Limitation.

Returns
The restriction of this Limitation.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
dict

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()

Returns the string representation of the model.

Return type
`str`

property type: `str`

Gets the type of this Limitation.

Returns
The type of this Limitation.

Return type
str

isogeo_pysdk.models.link module

Isogeo API v1 - Model of Link entity

See: http://help.isogeo.com/api/complete/index.html#definition-link

class isogeo_pysdk.models.link.Link(*_id=None*, *actions=None*, *fileName=None*, *kind=None*, *link=None*, *size=None*, *title=None*, *type=None*, *url=None*, *parent_resource=None*)

Bases: `object`

Links are entities included as subresource into metadata for data history title.

Example

```
{
    '_id': string (uuid),
    'actions': list,
    'fileName': string,
    'kind': string,
    'parent_resource': string (uuid),
    'size': int,
    'title': string,
    'type': string,
    'url': string
}
```

ATTR_CREA = {'actions': <class 'list'>, 'fileName': <class 'str'>, 'kind': <class 'str'>, 'link': <class 'dict'>, 'parent_resource': <class 'str'>, 'title': <class 'str'>, 'type': <class 'str'>, 'url': <class 'str'>}

ATTR_MAP = {}

ATTR_TYPES = {'_id': <class 'str'>, 'actions': <class 'list'>, 'fileName': <class 'str'>, 'kind': <class 'str'>, 'link': <class 'dict'>, 'parent_resource': <class 'str'>, 'size': <class 'int'>, 'title': <class 'str'>, 'type': <class 'str'>, 'url': <class 'str'>}

property actions: `list`

Gets the actions of this Link.

Returns
The actions of this Link.

Return type
list

property fileName: list

Gets the fileName of this Link.

Returns
The fileName of this Link.

Return type
list

property kind: str

Gets the kind of this Link.

Returns
The kind of this Link.

Return type
str

property link: dict

Gets the associated link of this Link.

Returns
The associated link of this Link.

Return type
dict

property size: int

Gets the size of the hosted data.

Returns
The size of the hosted data.

Return type
int

property title: str

Gets the title of this Link.

Returns
The title of this Link.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
`dict`

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
`dict`

to_str()

Returns the string representation of the model.

Return type
`str`

`property type: str`

Gets the type of this Link.

Returns
The type of this Link.

Return type
str

`property url: str`

Gets the url of this Link.

Return type
`str`

Returns
The url of this Link.

Rurl
str

isogeo_pysdk.models.metadata module

Isogeo API v1 - Model of Metadata (= Resource) entity

See: http://help.isogeo.com/api/complete/index.html#definition-resource

`class isogeo_pysdk.models.metadata.Metadata(`*_abilities=None*, *_created=None*, *_creator=None*, *_id=None*, *_modified=None*, *abstract=None*, *collectionContext=None*, *collectionMethod=None*, *conditions=None*, *contacts=None*, *coordinateSystem=None*, *created=None*, *distance=None*, *editionProfile=None*, *encoding=None*, *envelope=None*, *events=None*, *featureAttributes=None*, *features=None*, *format=None*, *formatVersion=None*, *geoContext=None*, *geometry=None*, *keywords=None*, *language=None*, *layers=None*, *limitations=None*, *links=None*, *modified=None*, *name=None*, *operations=None*, *path=None*, *precision=None*, *published=None*, *scale=None*, *series=None*, *serviceLayers=None*, *specifications=None*, *tags=None*, *thumbnailUrl=None*, *title=None*, *topologicalConsistency=None*, *type=None*, *updateFrequency=None*, *validFrom=None*, *validTo=None*, *validityComment=None*, ***kwargs*`)`

Bases: `object`

Metadata are the main entities in Isogeo.

Example

```
{
    "_abilities": [
        "string"
    ],
    "_created": "string (date-time)",
    "_creator": {
        "_abilities": [
```

(continues on next page)

(continued from previous page)

```
        "string"
        ],
        "_created": "string (date-time)",
        "_id": "string (uuid)",
        "_modified": "string (date-time)",
        "areKeywordsRestricted": "boolean",
        "canCreateMetadata": "boolean",
        "code": "string",
        "contact": {
        "_created": "string (date-time)",
        "_id": "string (uuid)",
        "_modified": "string (date-time)",
        "addressLine1": "string",
        "addressLine2": "string",
        "addressLine3": "string",
        "available": "string",
        "city": "string",
        "count": "integer (int32)",
        "countryCode": "string",
        "email": "string",
        "fax": "string",
        "hash": "string",
        "name": "string",
        "organization": "string",
        "phone": "string",
        "type": "string",
        "zipCode": "string"
        },
        "keywordsCasing": "string",
        "metadataLanguage": "string",
        "themeColor": "string"
    },
    "_id": "string (uuid)",
    "_modified": "string (date-time)",
    "abstract": "string",
    "bbox": [
        "number (double)"
    ],
    "collectionContext": "string",
    "collectionMethod": "string",
    "conditions": [
        {
        "_id": "string (uuid)",
        "description": "string",
        "license": {
            "_id": "string (uuid)",
            "content": "string",
            "count": "integer (int32)",
            "link": "string",
            "name": "string"
        }
        }
```

(continues on next page)

(continued from previous page)

```
    ],
    "contacts": [
        {
        "_id": "string (uuid)",
        "contact": {
            "_created": "string (date-time)",
            "_id": "string (uuid)",
            "_modified": "string (date-time)",
            "addressLine1": "string",
            "addressLine2": "string",
            "addressLine3": "string",
            "available": "string",
            "city": "string",
            "count": "integer (int32)",
            "countryCode": "string",
            "email": "string",
            "fax": "string",
            "hash": "string",
            "name": "string",
            "organization": "string",
            "phone": "string",
            "type": "string",
            "zipCode": "string"
        },
        "role": "string"
        }
    ],
    "context": "object",
    "coordinate-system": "object",
    "created": "string (date-time)",
    "distance": "number (double)",
    "editionProfile": "string",
    "encoding": "string",
    "envelope": "object",
    "features": "integer (int32)",
    "format": "string",
    "formatVersion": "string",
    "geometry": "string",
    "height": "integer (int32)",
    "keywords": [
        {}
    ]
}
```

`ATTR_CREA = {'abstract': <class 'str'>, 'collectionContext': <class 'str'>, 'collectionMethod': <class 'str'>, 'distance': <class 'float'>, 'editionProfile': <class 'str'>, 'encoding': <class 'str'>, 'envelope': <class 'dict'>, 'features': <class 'int'>, 'format': <class 'str'>, 'formatVersion': <class 'str'>, 'geoContext': <class 'str'>, 'geometry': <class 'str'>, 'language': <class 'str'>, 'name': <class 'str'>, 'path': <class 'str'>, 'precision': <class 'str'>, 'scale': <class 'int'>, 'series': <class 'bool'>, 'title': <class 'str'>, 'topologicalConsistency': <class 'str'>, 'type': <class 'str'>, 'updateFrequency': <class 'str'>, 'validFrom': <class 'str'>, 'validTo': <class 'str'>, 'validityComment': <class 'str'>}`

`ATTR_MAP = {'coordinateSystem': 'coordinate-system', 'featureAttributes': 'feature-attributes'}`

`ATTR_TYPES = {'_abilities': <class 'list'>, '_created': <class 'str'>, '_creator': <class 'dict'>, '_id': <class 'str'>, '_modified': <class 'str'>, 'abstract': <class 'str'>, 'collectionContext': <class 'str'>, 'collectionMethod': <class 'str'>, 'conditions': <class 'list'>, 'contacts': <class 'list'>, 'coordinateSystem': <class 'dict'>, 'created': <class 'str'>, 'distance': <class 'float'>, 'editionProfile': <class 'str'>, 'encoding': <class 'str'>, 'envelope': <class 'dict'>, 'events': <class 'list'>, 'featureAttributes': <class 'list'>, 'features': <class 'int'>, 'format': <class 'str'>, 'formatVersion': <class 'str'>, 'geoContext': <class 'str'>, 'geometry': <class 'str'>, 'keywords': <class 'list'>, 'language': <class 'str'>, 'layers': <class 'list'>, 'limitations': <class 'list'>, 'links': <class 'list'>, 'modified': <class 'str'>, 'name': <class 'str'>, 'operations': <class 'list'>, 'path': <class 'str'>, 'precision': <class 'str'>, 'published': <class 'str'>, 'scale': <class 'int'>, 'series': <class 'bool'>, 'serviceLayers': <class 'list'>, 'specifications': <class 'list'>, 'tags': <class 'list'>, 'thumbnailUrl': <class 'str'>, 'title': <class 'str'>, 'topologicalConsistency': <class 'str'>, 'type': <class 'str'>, 'updateFrequency': <class 'str'>, 'validFrom': <class 'str'>, 'validTo': <class 'str'>, 'validityComment': <class 'str'>}`

`property abstract: str`

Gets the abstract.

Returns
The abstract of this Metadata.

Return type
str

`admin_url`(*url_base='https://app.isogeo.com'*)

Returns the administration URL (https://app.isogeo.com) for this metadata.

Parameters
`url_base` (*str*) – base URL of admin site. Defaults to: https://app.isogeo.com

Return type
str

`classmethod clean_attributes`(*raw_object*)

Renames attributes which are incompatible with Python (hyphens…). See related issue: https://github.com/isogeo/isogeo-api-py-minsdk/issues/82.

Parameters
`raw_object` (*dict*) – metadata dictionary returned by a request.json()

Returns
the metadata with correct attributes

Return type
Metadata

property collectionContext: str

Gets the collectionContext of this Metadata.

Returns
The collectionContext of this Metadata.

Return type
str

property collectionMethod: str

Gets the collection method of this Metadata.

Returns
The collection method of this Metadata.

Return type
str

property conditions: list

Gets the conditions of this Metadata.

Returns
The conditions of this Metadata.

Return type
list

property contacts: list

Gets the contacts of this Metadata.

Returns
The contacts of this Metadata.

Return type
list

property coordinateSystem: *CoordinateSystem*

Gets the coordinateSystem of this Metadata.

Returns
The coordinateSystem of this Metadata.

Return type
CoordinateSystem

property created: str

Gets the creation date of the data described by the Metadata. It's the equivalent of the *created* original attribute (renamed to avoid conflicts with the _created` one).

Date format is: *%Y-%m-%dT%H:%M:%S+00:00*.

Returns
The creation of this Metadata.

Return type
str

property distance: str

Gets the distance of this Metadata.

Returns
The distance of this Metadata.

Return type
str

property editionProfile: str

Gets the editionProfile of this Metadata.

Returns
The editionProfile of this Metadata.

Return type
str

property encoding: str

Gets the encoding of this Metadata.

Returns
The encoding of this Metadata.

Return type
str

property envelope: str

Gets the envelope of this Metadata.

Returns
The envelope of this Metadata.

Return type
str

property events: list

Gets the events of this Metadata.

Returns
The events of this Metadata.

Return type
list

property featureAttributes: list

Gets the featureAttributes of this Metadata.

Returns
The featureAttributes of this Metadata.

Return type
list

property features: int

Gets the features of this Metadata.

Returns
The features of this Metadata.

Return type
int

property format: str

Gets the format of this Metadata.

Returns
The format of this Metadata.

Return type
str

property formatVersion: str

Gets the formatVersion of this Metadata.

Returns
The formatVersion of this Metadata.

Return type
str

property geoContext: str

Gets the geoContext of this Metadata.

Returns
The geoContext of this Metadata.

Return type
str

property geometry: str

Gets the geometry of this Metadata.

Returns
The geometry of this Metadata.

Return type
str

property groupId: str

Shortcut to get the UUID of the workgroup which owns the Metadata.

Return type
`str`

property groupName: str

Shortcut to get the name of the workgroup which owns the Metadata.

Return type
`str`

property keywords: str

Gets the keywords of this Metadata.

Returns
The keywords of this Metadata.

Return type
str

property language: str

Gets the language of this Metadata.

Returns
The language of this Metadata.

Return type
str

`property layers: list`

Gets the layers of this Metadata.

Returns
The layers of this Metadata.

Return type
list

`property limitations: str`

Gets the limitations of this Metadata.

Returns
The limitations of this Metadata.

Return type
str

`property links: str`

Gets the links of this Metadata.

Returns
The links of this Metadata.

Return type
str

`property modified: str`

Gets the last modification date of the data described by this Metadata.

It's the equivalent of the *created* original attribute (renamed to avoid conflicts with the _created` one).

Returns
The modification of this Metadata.

Return type
str

`property name: str`

Gets the name of this Metadata.

Returns
The name of this Metadata.

Return type
str

`property operations: list`

Gets the operations of this Metadata.

Returns
The operations of this Metadata.

Return type
list

`property path: str`

Gets the path of this Metadata.

Returns
The path of this Metadata.

Return type
str

property precision: str

Gets the precision of this Metadata.

Returns
The precision of this Metadata.

Return type
str

property published: str

Gets the published of this Metadata.

Returns
The published of this Metadata.

Return type
str

property scale: str

Gets the scale of this Metadata.

Returns
The scale of this Metadata.

Return type
str

property series: str

Gets the series of this Metadata.

Returns
The series of this Metadata.

Return type
str

property serviceLayers: list

Gets the serviceLayers of this Metadata.

Returns
The serviceLayers of this Metadata.

Return type
list

signature(*included_attributes=('coordinateSystem', 'envelope', 'features', 'featureAttributes', 'format', 'geometry', 'groupId', 'name', 'path', 'series', 'title', 'type')*)

Calculate a hash cumulating certain attributes values. Useful to Scan or comparison operations.

Parameters
included_attributes (*tuple*) – object attributes to include in hash. Default: {("coordinateSystem","envelope","features","featuresAttributes","format","geometry","groupId","name","path","series","ti

Return type
str

property specifications: str

Gets the specifications of this Metadata.

Returns
The specifications of this Metadata.

Return type
str

property tags: str

Gets the tags of this Metadata.

Returns
The tags of this Metadata.

Return type
str

property thumbnailUrl: str

Gets the thumbnailUrl of this Metadata.

Returns
The thumbnailUrl of this Metadata.

Return type
str

property title: str

Gets the title of this Metadata.

Returns
The title of this Metadata.

Return type
str

title_or_name(*slugged=False*)

Gets the title of this Metadata or the name if there is no title. It can return a slugified value.

Parameters
slugged (*bool*) – slugify title. Defaults to *False*.

Returns
the title or the name of this Metadata.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
dict

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()

Returns the string representation of the model.

Return type
str

property topologicalConsistency: str

Gets the topologicalConsistency of this Metadata.

Returns
The topologicalConsistency of this Metadata.

Return type
str

property type: str

Gets the type of this Metadata.

Returns
The type of this Metadata.

Return type
str

property typeFilter: str

Shortcut to get the type as expected in search filter.

Return type
str

property updateFrequency: str

Gets the updateFrequency of this Metadata.

Returns
The updateFrequency of this Metadata.

Return type
str

property validFrom: str

Gets the validFrom of this Metadata.

Returns
The validFrom of this Metadata.

Return type
str

property validTo: str

Gets the validTo of this Metadata.

Returns
The validTo of this Metadata.

Return type
str

property validityComment: str

Gets the validityComment of this Metadata.

Returns
The validityComment of this Metadata.

Return type
str

isogeo_pysdk.models.metadata_search module

Isogeo API v1 - Model of Metadata search entity

See: http://help.isogeo.com/api/complete/index.html#definition-search

class isogeo_pysdk.models.metadata_search.**MetadataSearch**(*envelope=None*, *limit=None*, *offset=None*, *query=None*, *results=None*, *tags=None*, *total=None*)

Bases: `object`

Metadata searchs are entities used to organize and shares metadata of a workgroup.

ATTR_TYPES = {'envelope': <class 'object'>, 'limit': <class 'int'>, 'offset': <class 'int'>, 'query': <class 'dict'>, 'results': <class 'list'>, 'tags': <class 'dict'>, 'total': <class 'int'>}

property envelope: dict

Gets the abilities of this Metadata search.

Returns
The abilities of this Metadata search.

Return type
dict

property limit: int

Gets the created of this Metadata search.

Returns
The created of this Metadata search.

Return type
str

property offset: int

Gets the offset of this Metadata search.

Returns
The offset of this Metadata search.

Return type
int

property query: dict

Gets the modified of this Metadata search.

Returns
The modified of this Metadata search.

Return type
dict

property results: list

Gets the tag of this Metadata search.

Returns
The tag of this Metadata search.

Return type
list

`property tags: dict`
Gets the tags of this Metadata search.

Returns
The tags of this Metadata search.

Return type
dict

`to_dict()`
Returns the model properties as a dict.

Return type
`dict`

`to_str()`
Returns the string representation of the model.

Return type
`str`

`property total: int`
Gets the total of this Metadata search.

Returns
The total of this Metadata search.

Return type
int

isogeo_pysdk.models.service_layer module

Isogeo API v1 - Model of ServiceLayer entity

See: http://help.isogeo.com/api/complete/index.html#definition-serviceLayer

`class isogeo_pysdk.models.service_layer.ServiceLayer(`*_id=None*, *dataset=None*, *datasets=None*, *id=None*, *name=None*, *mimeTypes=None*, *titles=None*, *title=None*, *parent_resource=None*`)`

Bases: `object`

ServiceLayers are entities defining rules of data creation.

Example

```
{
    "_id": "string (uuid)",
    "id": "string",
    "mimeTypes": [
        "string"
    ],
    "titles": [
```

(continues on next page)

(continued from previous page)

```
        {
            "lang": "string",
            "value": "string"
        }
    ],
    "title": "string"
}
```

ATTR_CREA = {'name': <class 'str'>, 'title': <class 'str'>, 'titles': <class 'list'>}

ATTR_MAP = {'name': 'id'}

ATTR_TYPES = {'_id': <class 'str'>, 'dataset': <class 'dict'>, 'datasets': <class 'list'>, 'mimeTypes': <class 'str'>, 'name': <class 'str'>, 'title': <class 'str'>, 'titles': <class 'list'>}

property dataset: dict

Gets the dataset used for Isogeo filters of this ServiceLayer.

Returns
The dataset of this ServiceLayer.

Return type
dict

property datasets: list

Gets the datasets used for Isogeo filters of this ServiceLayer.

Returns
The datasets of this ServiceLayer.

Return type
list

property mimeTypes: str

Gets the mimeTypes of this ServiceLayer.

Returns
The mimeTypes of this ServiceLayer.

Return type
str

property name: str

Gets the name used for Isogeo filters of this ServiceLayer.

Returns
The name of this ServiceLayer.

Return type
str

property title: str

Gets the title of this ServiceLayer.

Returns
The title of this ServiceLayer.

Return type
str

property titles: list

Gets the titles of this ServiceLayer.

Returns
The titles of this ServiceLayer.

Return type
list

to_dict()

Returns the model properties as a dict.

Return type
dict

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()

Returns the string representation of the model.

Return type
str

isogeo_pysdk.models.service_operation module

Isogeo API v1 - Model of ServiceOperation entity

See: http://help.isogeo.com/api/complete/index.html#definition-serviceOperation

class isogeo_pysdk.models.service_operation.**ServiceOperation**(*_id=None*, *mimeTypesIn=None*, *mimeTypesOut=None*, *name=None*, *url=None*, *verb=None*, *parent_resource=None*)

Bases: object

ServiceOperations are entities defining rules of data creation.

Example

```
{
    "_id": "string (uuid)",
    "mimeTypesIn": [
        "string"
    ],
    "mimeTypesOut": [
        "string"
    ],
    "name": "string",
    "url": "string",
    "verb": "string"
}
```

ATTR_CREA = {'name': <class 'str'>, 'verb': <class 'str'>}

ATTR_MAP = {}

ATTR_TYPES = {'_id': <class 'str'>, 'mimeTypesIn': <class 'list'>, 'mimeTypesOut': <class 'list'>, 'name': <class 'str'>, 'url': <class 'str'>, 'verb': <class 'str'>}

property mimeTypesIn: dict

Gets the mimeTypesIn used for Isogeo filters of this ServiceOperation.

Returns
The mimeTypesIn of this ServiceOperation.

Return type
dict

property mimeTypesOut: str

Gets the mimeTypesOut of this ServiceOperation.

Returns
The mimeTypesOut of this ServiceOperation.

Return type
str

property name: str

Gets the name used for Isogeo filters of this ServiceOperation.

Returns
The name of this ServiceOperation.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
dict

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()

Returns the string representation of the model.

Return type
str

property url: list

Gets the url of this ServiceOperation.

Returns
The url of this ServiceOperation.

Return type
list

property verb: list

Gets the verb of this ServiceOperation.

Returns
The verb of this ServiceOperation.

Return type
list

isogeo_pysdk.models.share module

Isogeo API v1 - Model of Share entity

See: http://help.isogeo.com/api/complete/index.html#definition-share

class isogeo_pysdk.models.share.Share(*_created=None*, *_creator=None*, *_id=None*, *_modified=None*, *applications=None*, *catalogs=None*, *groups=None*, *name=None*, *rights=None*, *type=None*, *urlToken=None*)

Bases: object

Shares are entities used to publish catalog(s) of metadata to applications.

Example

```
{
    "_created": "string (date-time)",
    "_creator": {
        "_abilities": [
        "string"
        ],
        "_created": "string (date-time)",
        "_id": "string (uuid)",
        "_modified": "string (date-time)",
        "areKeywordsRestricted": "boolean",
        "canCreateMetadata": "boolean",
        "catalogs": "string",
        "contact": {
        "_created": "string (date-time)",
        "_id": "string (uuid)",
        "_modified": "string (date-time)",
        "addressLine1": "string",
        "addressLine2": "string",
        "addressLine3": "string",
        "available": "string",
        "city": "string",
        "groups": "integer (int32)",
        "groupsryCode": "string",
        "email": "string",
        "fax": "string",
        "hash": "string",
        "name": "string",
        "organization": "string",
        "phone": "string",
        "type": "string",
        "zipCode": "string"
```

(continues on next page)

(continued from previous page)

```
        },
        "keywordsCasing": "string",
        "metadataLanguage": "string",
        "themeColor": "string"
    },
    "_id": "string (uuid)",
    "_modified": "string (date-time)",
    "applications": [
        {
        "_created": "string (date-time)",
        "_id": "string (uuid)",
        "_modified": "string (date-time)",
        "canHaveManyGroups": "boolean",
        "client_id": "string",
        "client_secret": "string",
        "groups": [
            {
            "_abilities": [
                "string"
            ],
            "_created": "string (date-time)",
            "_id": "string (uuid)",
            "_modified": "string (date-time)",
            "areKeywordsRestricted": "boolean",
            "canCreateMetadata": "boolean",
            "catalogs": "string",
            "contact": {
                "_created": "string (date-time)",
                "_id": "string (uuid)",
                "_modified": "string (date-time)",
                "addressLine1": "string",
                "addressLine2": "string",
                "addressLine3": "string",
                "available": "string",
                "city": "string",
                "groups": "integer (int32)",
                "groupsryCode": "string",
                "email": "string",
                "fax": "string",
                "hash": "string",
                "name": "string",
                "organization": "string",
                "phone": "string",
                "type": "string",
                "zipCode": "string"
            },
            "keywordsCasing": "string",
            "metadataLanguage": "string",
            "themeColor": "string"
            }
        ],
        "kind": "string",
```

(continues on next page)

(continued from previous page)

```
        "name": "string",
        "redirect_uris": [
            "string"
        ],
        "scopes": [
            "string"
        ],
        "staff": "boolean",
        "url": "string"
        }
    ],
    "catalogs": [
        {
        "$scan": "boolean",
        "_abilities": [
            "string"
        ],
        "_created": "string (date-time)",
        "_id": "string (uuid)",
        "_modified": "string (date-time)"
        }
    ]
}
```

`ATTR_CREA = {'name': <class 'str'>, 'rights': <class 'list'>, 'type': <class 'str'>}`

`ATTR_MAP = {}`

`ATTR_TYPES = {'_created': <class 'str'>, '_creator': <class 'isogeo_pysdk.models.workgroup.Workgroup'>, '_id': <class 'str'>, '_modified': <class 'str'>, 'applications': <class 'list'>, 'catalogs': <class 'list'>, 'groups': <class 'list'>, 'name': <class 'str'>, 'rights': <class 'list'>, 'type': <class 'str'>, 'urlToken': <class 'str'>}`

admin_url(*url_base='https://app.isogeo.com'*)

Returns the administration URL (https://app.isogeo.com) for this share.

Parameters
`url_base` (*str*) – base URL of admin site. Defaults to: https://app.isogeo.com

Return type
str

property applications: list

Gets the applications of this Share.

Returns
The applications of this Share.

Return type
str

property catalogs: list

Gets the catalogs of this Share.

Returns
The catalogs of this Share.

Return type
str

`property groups: list`

Gets the groups of this Share.

Returns
The groups of this Share.

Return type
list

`property name: str`

Gets the name of this Share.

Returns
The name of this Share.

Return type
str

`opencatalog_url`(*url_base='https://open.isogeo.com'*)

Returns the OpenCatalog URL for this share or None if OpenCatalog is not enabled.

Parameters
url_base (*str*) – base URL of OpenCatalog. Defaults to: https://open.isogeo.com

Return type
`str`

Returns

- False if the share type is not 'application'
- None if OpenCatalog is not enabled in the share
- URL of the OpenCatalog when everything is fine

`property rights: list`

Gets the rights of this Share.

Returns
The rights of this Share.

Return type
str

`to_dict()`

Returns the model properties as a dict.

Return type
`dict`

`to_dict_creation()`

Returns the model properties as a dict structured for creation purpose (POST)

Return type
`dict`

to_str()

Returns the string representation of the model.

Return type
`str`

property type: str

Gets the type of this Share.

Returns
The type of this Share.

Return type
str

property urlToken: str

Gets the urlToken of this Share.

Returns
The urlToken of this Share.

Return type
str

isogeo_pysdk.models.specification module

Isogeo API v1 - Model of Specification entity

See: http://help.isogeo.com/api/complete/index.html#definition-specification

class isogeo_pysdk.models.specification.**Specification**(*_abilities=None*, *_id=None*, *_tag=None*, *count=None*, *link=None*, *name=None*, *owner=None*, *published=None*)

Bases: `object`

Specifications are entities defining rules of data creation.

Example

```
{
    '_abilities': [],
    '_id': 'string (uuid)',
    '_tag': 'specification:isogeo:string (uuid)',
    'count': int,
    'link': string,
    'name': string,
    'published': '2016-06-30T00:00:00'
}
```

ATTR_CREA = {'link': <class 'str'>, 'name': <class 'str'>, 'published': <class 'str'>}

ATTR_MAP = {}

ATTR_TYPES = {'_abilities': <class 'str'>, '_id': <class 'str'>, '_tag': <class 'str'>, 'count': <class 'int'>, 'link': <class 'str'>, 'name': <class 'str'>, 'owner': <class 'dict'>, 'published': <class 'str'>}

property count: int

Gets the id of this Specification.

Returns
The id of this Specification.

Return type
str

property isLocked: bool

Shortcut to know if the Specification is owned by Isogeo or a workgroup.

Return type
bool

Returns

- None if tag is None too
- True if the specification is owned by Isogeo = locked
- False if the specification is owned by a workgroup = not locked

property link: str

Gets the link (URL) of this Specification.

Returns
The link (URL) of this Specification.

Return type
str

property name: str

Gets the id of this Specification.

Returns
The id of this Specification.

Return type
str

property owner

Gets the owner of this Specification.

Returns
The owner of this Specification.

Return type
Workgroup

property published: str

Gets the zip (postal) code of this Specification.

Returns
The zip (postal) code of this Specification.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
`dict`

to_dict_creation()
Returns the model properties as a dict structured for creation purpose (POST)

Return type
`dict`

to_str()
Returns the string representation of the model.

Return type
`str`

isogeo_pysdk.models.thesaurus module

Isogeo API v1 - Model of Thesaurus entity

See: http://help.isogeo.com/api/complete/index.html#definition-thesaurus

`class isogeo_pysdk.models.thesaurus.`**`Thesaurus`**`(`*_abilities=None*, *_id=None*, *code=None*, *name=None*`)`

Bases: `object`

Thesaurus are entities which can be used in shares.

Example

```
{
    '_abilities': [],
    '_id': '926f969ee2bb470a84066625f68b96bb',
    'code': 'iso19115-topic',
    'name': 'MD_TopicCategoryCode'
}
```

`ATTR_CREA = {'name': <class 'str'>}`

`ATTR_MAP = {}`

`ATTR_TYPES = {'_abilities': <class 'list'>, '_id': <class 'str'>, 'code': <class 'str'>, 'name': <class 'str'>}`

`property code: str`
Gets the code of this Thesaurus.

Returns
The code of this Thesaurus.

Return type
str

`property name: str`
Gets the name of this Thesaurus.

Returns
The name of this Thesaurus.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
dict

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()

Returns the string representation of the model.

Return type
str

isogeo_pysdk.models.user module

Isogeo API v1 - Model of User entity

See: http://help.isogeo.com/api/complete/index.html#definition-user

class isogeo_pysdk.models.user.**User**(*_abilities=None*, *_created=None*, *_id=None*, *_modified=None*, *contact=None*, *language=None*, *mailchimp=None*, *memberships=None*, *staff=None*, *timezone=None*)

Bases: object

Users in Isogeo platform.

Example

```
{
    "_abilities": [
        "string"
    ],
    "_created": "string (date-time)",
    "_id": "string (uuid)",
    "_modified": "string (date-time)",
    "contact": {
        "_created": "string (date-time)",
        "_id": "string (uuid)",
        "_modified": "string (date-time)",
        "addressLine1": "string",
        "addressLine2": "string",
        "addressLine3": "string",
        "available": "string",
        "city": "string",
        "count": "integer (int32)",
        "countryCode": "string",
        "email": "string",
        "fax": "string",
        "hash": "string",
        "name": "string",
        "organization": "string",
```

(continues on next page)

(continued from previous page)

```
        "phone": "string",
        "type": "string",
        "zipCode": "string"
    },
    "language": "string",
    "staff": "boolean",
    "timezone": "string"
}
```

`ATTR_CREA = {'contact': <class 'isogeo_pysdk.models.contact.Contact'>, 'language': <class 'str'>, 'mailchimp': <class 'str'>, 'staff': <class 'bool'>, 'timezone': <class 'str'>}`

`ATTR_MAP = {}`

`ATTR_TYPES = {'_abilities': <class 'list'>, '_created': <class 'str'>, '_id': <class 'str'>, '_modified': <class 'str'>, 'contact': <class 'isogeo_pysdk.models.contact.Contact'>, 'language': <class 'str'>, 'mailchimp': <class 'dict'>, 'staff': <class 'bool'>, 'timezone': <class 'str'>}`

`property contact:` *Contact*

Gets the contact of this user.

Returns
The contact of this user.

Return type
Contact

`property email: str`

Shortcut to get the email from the contact data linked to the user.

Return type
`str`

`property language: str`

Gets the id of this User.

Returns
The id of this User.

Return type
str

`property mailchimp: str`

Gets the id of this User.

Returns
The second address line of this User.

Return type
str

`property name: str`

Shortcut to get the name from the contact data linked to the user.

Return type
`str`

property staff: bool

Staff status for the User.

Returns
the staff status of the User

Return type
bool

property timezone: str

Gets the timezone of this User.

Returns
The timezone of this User.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
`dict`

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
`dict`

to_str()

Returns the string representation of the model.

Return type
`str`

isogeo_pysdk.models.workgroup module

Isogeo API v1 - Model of Workgroup entity

See: http://help.isogeo.com/api/complete/index.html#definition-workgroup

class isogeo_pysdk.models.workgroup.**Workgroup**(*_abilities=None*, *_created=None*, *_id=None*, *_modified=None*, *_tag=None*, *areKeywordsRestricted=None*, *canCreateLegacyServiceLinks=None*, *canCreateMetadata=None*, *code=None*, *contact=None*, *hasCswClient=None*, *hasScanFme=None*, *keywordsCasing=None*, *limits=None*, *metadataLanguage=None*, *themeColor=None*)

Bases: `object`

Workgroups are entities containing metadata.

Example

```
{
    '_abilities': [
        'group:manage',
        'group:update'
    ],
    '_created': '2015-05-21T12:08:16.4295098+00:00',
    '_id': '32f7e95ec4e94ca3bc1afda960003882',
    '_modified': '2018-12-27T10:47:28.7880956+00:00',
    '_tag': 'owner:32f7e95ec4e94ca3bc1afda960003882',
    'areKeywordsRestricted': False,
    'canCreateLegacyServiceLinks': True,
    'canCreateMetadata': True,
    'contact': {
        '_deleted': False,
        '_id': '2a3aefc4f80347f590afe58127f6cb0f',
        '_tag': 'contact:group:2a3aefc4f80347f590afe58127f6cb0f',
        'addressLine1': '26 rue du faubourg Saint-Antoine',
        'addressLine2': '4éme étage',
        'addressLine3': 'bouton porte',
        'available': False,
        'city': 'Paris',
        'countryCode': 'FR',
        'email': 'dev@isogeo.com',
        'fax': '33 (0)9 67 46 50 06',
        'name': 'Isogeo Test',
        'phone': '33 (0)9 67 46 50 06',
        'type': 'group',
        'zipCode': '75012'
    },
    'hasCswClient': True,
    'hasScanFme': True,
    'keywordsCasing': 'lowercase',
    'limits': {
        'canDiffuse': False,
        'canShare': True,
        'Workgroups': {
            'current': 1,
            'max': -1
        },
        'resources': {
            'current': 2,
            'max': 20
        },
        'upload': {
            'current': 0,
            'max': 1073741824
        },
        'users': {
            'current': 1,
            'max': 2}
        },
    'metadataLanguage': 'fr',
    'themeColor': '#4499A1'
```

(continues on next page)

(continued from previous page)

```
}
```

ATTR_CREA = {'canCreateLegacyServiceLinks': <class 'bool'>, 'canCreateMetadata': <class 'bool'>, 'contact': <class 'isogeo_pysdk.models.contact.Contact'>, 'metadataLanguage': <class 'str'>}

ATTR_MAP = {'contact': ['contact.addressLine1', 'contact.addressLine2', 'contact.addressLine3', 'contact.city', 'contact.countryCode', 'contact.email', 'contact.fax', 'contact.name', 'contact.phone', 'contact.zipCode']}

ATTR_TYPES = {'_abilities': <class 'list'>, '_created': <class 'str'>, '_id': <class 'str'>, '_modified': <class 'str'>, '_tag': <class 'str'>, 'areKeywordsRestricted': <class 'bool'>, 'canCreateLegacyServiceLinks': <class 'bool'>, 'canCreateMetadata': <class 'bool'>, 'code': <class 'str'>, 'contact': <class 'isogeo_pysdk.models.contact.Contact'>, 'hasCswClient': <class 'bool'>, 'hasScanFme': <class 'bool'>, 'keywordsCasing': <class 'str'>, 'limits': <class 'dict'>, 'metadataLanguage': <class 'str'>, 'themeColor': <class 'str'>}

admin_url(*url_base='https://app.isogeo.com'*)

Returns the administration URL (https://app.isogeo.com) for this group.

Parameters

url_base (*str*) – base URL of admin site. Defaults to: https://app.isogeo.com. Can also be https://manage.isogeo.com.

Return type

str

property areKeywordsRestricted: str

Gets the areKeywordsRestricted of this Workgroup.

Returns

The areKeywordsRestricted of this Workgroup.

Return type

str

property canCreateLegacyServiceLinks: str

Gets the canCreateLegacyServiceLinks of this Workgroup.

Returns

The canCreateLegacyServiceLinks of this Workgroup.

Return type

str

property canCreateMetadata: bool

Gets the canCreateMetadata of this Workgroup.

Returns

The canCreateMetadata of this Workgroup.

Return type

str

property code: str

Gets the code of this Workgroup.

Returns
The code of this Workgroup.

Return type
str

property contact: *Contact*

Gets the contact of this Workgroup.

Returns
The contact of this Workgroup.

Return type
dict

property hasCswClient: bool

Gets the hasCswClient of this Workgroup.

Returns
The hasCswClient of this Workgroup.

Return type
bool

property hasScanFme: bool

Find out if the group has access to the Scan.

Returns
The hasScanFme value of this Workgroup.

Return type
bool

property keywordsCasing: str

Gets the keywordsCasing of this Workgroup.

Returns
The keywordsCasing of this Workgroup.

Return type
str

property limits: dict

Gets the limits of this Workgroup.

Returns
The limits of this Workgroup.

Return type
dict

property metadataLanguage: str

Gets the metadataLanguage of this Workgroup.

Returns
The metadataLanguage of this Workgroup.

Return type
str

property name: str

Shortcut to get the name of the workgroup.

Return type
str

property themeColor: str

Gets the themeColor of this Workgroup.

Returns
The themeColor of this Workgroup.

Return type
str

to_dict()

Returns the model properties as a dict.

Return type
dict

to_dict_creation()

Returns the model properties as a dict structured for creation purpose (POST)

Return type
dict

to_str()

Returns the string representation of the model.

Return type
str

Submodules

isogeo_pysdk.api_hooks module

Complementary set of hooks to use with Isogeo API.

class isogeo_pysdk.api_hooks.IsogeoHooks

Bases: object

Custom requests event hooks for Isogeo API.

Requests has a hook system that you can use to manipulate portions of the request process, or signal event handling. This module is a set of custom hooks to handle Isogeo API responses.

WIP

autofix_attributes_resource(*resp*, **args*, ***kwargs*)

check_for_error(*resp*, **args*, ***kwargs*)

isogeo_pysdk.checker module

Complementary set of tools to make some checks on requests to Isogeo API.

class isogeo_pysdk.checker.**IsogeoChecker**

Bases: `object`

Complementary set of tools to make some checks on requests to Isogeo API.

check_api_response(*response*)

Check API response and raise exceptions if needed.

Parameters
response (*requests.models.Response*) – request response to check

Return type
True or tuple

Example

```
>>> checker.check_api_response(<Response [500]>)
(False, 500)
```

check_edit_tab(*tab*, *md_type*)

Check if asked tab is part of Isogeo web form and reliable with metadata type.

Parameters

- **tab** (*str*) – tab to check. Must be one one of EDIT_TABS attribute
- **md_type** (*str*) – metadata type. Must be one one of FILTER_TYPES

check_internet_connection(*remote_server='api.isogeo.com'*, *proxies=None*)

Test if an internet connection is operational. Src: https://stackoverflow.com/a/20913928/2556577.

Parameters
remote_server (*str*) – remote server used to check

Return type
`bool`

classmethod check_is_uuid(*uuid_str*)

Check if it's an Isogeo UUID handling specific form.

Parameters
uuid_str (*str*) – UUID string to check

check_request_parameters(*parameters={}*)

Check parameters passed to avoid errors and help debug.

Parameters
response (*dict*) – search request parameters

isogeo_pysdk.decorators module

Isogeo Python SDK - Decorators

class `isogeo_pysdk.decorators.`**`ApiDecorators`**

Bases: `object`

`api_client = None`

isogeo_pysdk.exceptions module

Isogeo Python SDK - Custom exceptions

See: https://docs.python.org/fr/3/tutorial/errors.html#user-defined-exceptions

exception `isogeo_pysdk.exceptions.`**`AlreadyExistError`**

Bases: *`IsogeoSdkError`*

An object with similar properties already exists in Isogeo database.

exception `isogeo_pysdk.exceptions.`**`IsogeoSdkError`**

Bases: `Exception`

Base class for exceptions in Isogeo Python SDK package.

isogeo_pysdk.isogeo module

Python SDK wrapping the Isogeo API.

Author: Julien Moura (@geojulien) for @Isogeo

class `isogeo_pysdk.isogeo.`**`Isogeo`**(*auth_mode='group'*, *client_secret=None*, *platform='qa'*, *proxy=None*, *timeout=(15, 45)*, *lang='fr'*, *app_name='isogeo-pysdk/3.8.0'*, *max_retries=2*, *pool_connections=20*, *pool_maxsize=50*, *isogeo_urls={}*, ***kwargs*)

Bases: `OAuth2Session`

Main class in Isogeo API Python wrapper. Manage authentication and requests to the REST API. Inherits from `requests_oauthlib.OAuth2Session`.

Inherited:

Parameters

- **`client_id`** (*`str`*) – Client id obtained during registration
- **`redirect_uri`** (*`str`*) – Redirect URI you registered as callback
- **`auto_refresh_url`** (*`list`*) – Refresh token endpoint URL, must be HTTPS. Supply this if you wish the client to automatically refresh your access tokens.

Package specific:

Parameters

- **`client_secret`** (*`str`*) – application oAuth2 secret
- **`auth_mode`** (*`str`*) – oAuth2 authentication flow to use. Must be one of 'AUTH_MODES'
- **`platform`** (*`str`*) – to request production or quality assurance

- **proxy** (*dict*) – dictionary of proxy settings as described in Requests
- **lang** (*str*) – API localization ("en" or "fr"). Defaults to 'fr'.
- **app_name** (*str*) – to custom the application name and user-agent
- **max_retries** (*int*) – custom the maximum number of retries each connection should attempt. See: Requests
- **pool_connections** (*int*) – custom the number of urllib3 connection pools to cache. See: Requests
- **pool_maxsize** (*int*) – custom the maximum number of connections to save in the pool. See: Requests
- **isogeo_urls** (*dict*) – Only needed when platform is "custom", a dictionnary of specific Isogeo URLs.

Returns
authenticated requests Session you can use to send requests to the API.

Return type
requests_oauthlib.OAuth2Session

Example

```python
# using oAuth2 Password Credentials Grant (Legacy Application)
#  (for scripts executed on the server-side with user credentials
#  but without requiring user action)
isogeo = Isogeo(
    client_id=environ.get("ISOGEO_API_USER_LEGACY_CLIENT_ID"),
    client_secret=environ.get("ISOGEO_API_USER_LEGACY_CLIENT_SECRET"),
    auth_mode="user_legacy",
    auto_refresh_url="{}/oauth/token".format(environ.get("ISOGEO_ID_URL")),
    platform=environ.get("ISOGEO_PLATFORM", "qa"),
)

# getting a token
isogeo.connect(
    username=environ.get("ISOGEO_USER_NAME"),
    password=environ.get("ISOGEO_USER_PASSWORD"),
)

# using oAuth2 Client Credentials Grant (Backend Application)
#  (for scripts executed on the server-side with only application credentials
#  but limited to read-only in Isogeo API)
isogeo = Isogeo(
    client_id=environ.get("ISOGEO_API_DEV_ID"),
    client_secret=environ.get("ISOGEO_API_DEV_SECRET"),
    auth_mode="group",
    auto_refresh_url="{}/oauth/token".format(environ.get("ISOGEO_ID_URL")),
    platform=environ.get("ISOGEO_PLATFORM", "qa"),
)

# getting a token
isogeo.connect()
```

AUTH_MODES = {'group': {'client_id': <class 'str'>, 'client_secret': <class 'str'>}, 'guess': {}, 'user_legacy': {'auto_refresh_url': <class 'str'>, 'client_id': <class 'str'>, 'client_secret': <class 'str'>}, 'user_private': {'auto_refresh_url': <class 'str'>, 'client_id': <class 'str'>, 'client_secret': <class 'str'>, 'redirect_uris': <class 'list'>}, 'user_public': {'auto_refresh_url': <class 'str'>, 'client_id': <class 'str'>, 'client_secret': <class 'str'>, 'redirect_uris': <class 'list'>}}

PLATFORM_VALUES = ['qa', 'prod', 'custom']

connect(*username=None*, *password=None*)

Custom the HTTP client and authenticate application with user credentials and fetch token.

Isogeo API uses oAuth 2.0 protocol (https://tools.ietf.org/html/rfc6749) see: http://help.isogeo.com/api/fr/authentication/concepts.html

Parameters

- **username** (*str*) – user login (email). Not required for group apps (Client Credentials).
- **password** (*str*) – user password. Not required for group apps (Client Credentials).

classmethod guess_auth_mode()

property header: dict

Return type
dict

isogeo_pysdk.translator module

Additionnal strings to be translated from Isogeo API.

class isogeo_pysdk.translator.**IsogeoTranslator**(*lang='FR'*)

Bases: object

Makes easier the translation of Isogeo API specific strings.

Parameters
lang (*str*) – language code to apply. EN or FR.

tr(*subdomain*, *string_to_translate=''*)

Returns translation of string passed.

Parameters

- **subdomain** (*str*) – subpart of strings dictionary. Must be one of self.translations.keys() i.e. 'restrictions'
- **string_to_translate** (*str*) – string you want to translate

Return type
str

isogeo_pysdk.type_hints_custom module

Custom type hints used in the SDK

isogeo_pysdk.utils module

Complementary set of utils to use with Isogeo API.

class isogeo_pysdk.utils.**IsogeoUtils**(*proxies={}*)

Bases: `object`

Complementary set of utilitary methods and functions to make it easier using Isogeo API.

API_URLS = {'prod': 'v1.api.isogeo.com', 'qa': 'v1.api.qa.isogeo.com'}

APP_URLS = {'prod': 'https://app.isogeo.com', 'qa': 'https://qa-isogeo-app.azurewebsites.net'}

CSW_URLS = {'prod': 'https://services.api.isogeo.com/', 'qa': 'http://services.api.qa.isogeo.com'}

MNG_URLS = {'prod': 'https://manage.isogeo.com', 'qa': 'https://qa-isogeo-manage.azurewebsites.net'}

OC_URLS = {'prod': 'https://open.isogeo.com', 'qa': 'https://qa-isogeo-open.azurewebsites.net'}

WEBAPPS = {'csw_getcap': {'args': ('share_id', 'share_token'), 'url': 'https://services.api.isogeo.com/ows/s/{share_id}/{share_token}?service=CSW&version=2.0.2&request=GetCapabilities'}, 'csw_getrec': {'args': ('md_uuid_urn', 'share_id', 'share_token'), 'url': 'https://services.api.isogeo.com/ows/s/{share_id}/{share_token}?service=CSW&version=2.0.2&request=GetRecordById&id={md_uuid_urn}&elementsetname=full&outputSchema=http://www.isotc211.org/2005/gmd'}, 'csw_getrecords': {'args': ('share_id', 'share_token'), 'url': 'https://services.api.isogeo.com/ows/s/{share_id}/{share_token}?service=CSW&version=2.0.2&request=GetRecords&ResultType=results&ElementSetName=brief&maxRecords=20&OutputFormat=applicatio xml&OutputSchema=http://www.opengis.net/cat/csw/2.0.2&namespace=xmlns(csw=http://www.opengis.net/cat/csw/2.0.2)&TypeNames=csw:Record&startPosition=1'}, 'oc': {'args': ('md_id', 'share_id', 'share_token'), 'url': 'https://{oc_url}/s/{share_id}/{share_token}/r/{md_id}'}, 'pixup_portal': {'args': ('md_id', 'portal_url'), 'url': 'http://{portal_url}/?muid={md_id}'}}

classmethod cache_clearer(*only_already_hit=1*)

Clear all LRU cached functions.

Parameters

only_already_hit (*bool*) – option to clear cache only for functions which have been already hit. Defaults to True.

classmethod convert_octets(*octets*)

Convert a mount of octets in readable size.

Parameters

octets (*int*) – mount of octets to convert

Example

```
>>> IsogeoUtils.convert_octets(1024)
"1ko"
```

Return type
`str`

classmethod convert_uuid(*in_uuid=<class 'str'>, mode=0*)

Convert a metadata UUID to its URI equivalent. And conversely.

Parameters

- **in_uuid** (*str*) – UUID or URI to convert
- **mode** (*int*) – conversion direction. Options:
  - 0 to HEX
  - 1 to URN (RFC4122)
  - 2 to URN (Isogeo specific style)

classmethod credentials_loader(*in_credentials='client_secrets.json'*)

Loads API credentials from a file, JSON or INI.

Parameters
in_credentials (*str*) – path to the credentials file. By default, *./client_secrets.json*

Return type
dict

Returns
a dictionary with credentials (ID, secret, URLs, platform…)

Example

```
api_credentials = IsogeoUtils.credentials_loader("./_auth/client_secrets.json")
pprint.pprint(api_credentials)
>>> {
        'auth_mode': 'group',
        'client_id': 'python-minimalist-sdk-test-uuid-1a2b3c4d5e6f7g8h9i0j11k12l
↪',
        'client_secret': 'application-secret-
↪1a2b3c4d5e6f7g8h9i0j11k12l13m14n15o16p17Q18rS',
        'kind': None,
        'platform': 'prod',
        'scopes': ['resources:read'],
        'staff': None,
        'type': None,
        'uri_auth': 'https://id.api.isogeo.com/oauth/authorize',
        'uri_base': 'https://api.isogeo.com',
        'uri_redirect': None,
        'uri_token': 'https://id.api.isogeo.com/oauth/token'
    }
```

classmethod encoded_words_to_text(*in_encoded_words*)

Pull out the character set, encoding, and encoded text from the input encoded words. Next, it decodes the encoded words into a byte string, using either the quopri module or base64 module as determined by the encoding. Finally, it decodes the byte string using the character set and returns the result.

See:

- https://github.com/isogeo/isogeo-api-py-minsdk/issues/32
- https://dmorgan.info/posts/encoded-word-syntax/

Parameters
`in_encoded_words` (*str*) – base64 or quori encoded character string.

`get_edit_url`(*metadata*, *tab='identification'*)

Returns the edition URL of a metadata.

Parameters

- **metadata** (`Metadata`) – metadata
- **tab** (*str*) – target tab in the web form. Optionnal. Defaults to 'identification'.

Return type
str

`get_isogeo_version`(*component='api'*, *prot='https'*)

Get Isogeo components versions. Authentication not required.

Parameters
component (*str*) – which platform component. Options:

- api [default]
- db
- app

`get_request_base_url`(*route*, *prot='https'*)

Build the request url for the specified route.

Parameters

- **route** (*str*) – route to format
- **prot** (*str*) – https [DEFAULT] or http

Return type
str

`classmethod get_url_base_from_url_token`(*url_api_token='https://id.api.isogeo.com/oauth/token'*)

Returns the Isogeo API root URL (not included into credentials file) from the token or the auth URL (always included).

Parameters
str (*url_api_token*) – url to Isogeo API ID token generator

Return type
str

Example

```
IsogeoUtils.get_url_base_from_url_token()
>>> "https://api.isogeo.com"
IsogeoUtils.get_url_base_from_url_token(url_api_token="https://id.api.qa.isogeo.
↪com/oauth/token")
>>> "https://api.qa.isogeo.com"
```

get_view_url(*webapp='oc'*, ***kwargs*)

Constructs the view URL of a metadata.

Parameters

- **webapp** (*str*) – web app destination
- **kwargs** (*dict*) – web app specific parameters. For example see WEBAPPS

classmethod guess_platform_from_url(*url='https://api.isogeo.com/'*)

Returns the Isogeo platform from a given URL.

Parameters

str (*url*) – URL string to guess from

Return type

str

Returns

"prod" or "qa" or "unknown"

Example

```
IsogeoUtils.guess_platform_from_url("https://api.isogeo.com")
>>> "prod"
IsogeoUtils.guess_platform_from_url("https://api.qa.isogeo.com")
>>> "qa"
IsogeoUtils.guess_platform_from_url("https://api.isogeo.ratp.local")
>>> "unknown"
```

classmethod hlpr_datetimes(*in_date*, *try_again=1*)

Helper to handle differnts dates formats. See: https://github.com/isogeo/isogeo-api-py-minsdk/issues/85

Parameters

- **raw_object** (*dict*) – metadata dictionary returned by a request.json()
- **try_again** (*bool*) – iterations on the method

Returns

a correct datetime object

Return type

datetime

Example

```
# for an event date
IsogeoUtils.hlpr_datetimes"2018-06-04T00:00:00+00:00")
>>> 2018-06-04 00:00:00
# for a metadata creation date with 6 digits as milliseconds
IsogeoUtils.hlpr_datetimes"2019-05-17T13:01:08.559123+00:00")
>>> 2019-05-17 13:01:08.559123
# for a metadata creation date with more than 6 digits as milliseconds
IsogeoUtils.hlpr_datetimes"2019-06-13T16:21:38.1917618+00:00")
>>> 2019-06-13 16:21:38.191761
```

lang = 'fr'

classmethod pages_counter(*total*, *page_size=100*)

Simple helper to handle pagination. Returns the number of pages for a given number of results.

Parameters

- **total** (*int*) – count of metadata in a search request
- **page_size** (*int*) – count of metadata to display in each page

Return type

int

register_webapp(*webapp_name*, *webapp_args*, *webapp_url*)

Register a new WEBAPP to use with the view URL builder.

Parameters

- **webapp_name** (*str*) – name of the web app to register
- **webapp_args** (*list*) – dynamic arguments to complete the URL. Typically 'md_id'.
- **webapp_url** (*str*) – URL of the web app to register with args tags to replace. Example: 'https://www.ppige-npdc.fr/portail/geocatalogue?uuid={md_id}'

set_base_url(*platform='prod'*, *dict_urls={}*)

Set Isogeo base URLs according to platform.

Parameters

- **platform** (*str*) – platform to use. Options:
  - prod [DEFAULT]
  - qa
  - custom
- **dict_urls** (*dict*) – Only needed when platform is "custom", a dictionnary of specific Isogeo URLs.

classmethod set_lang_and_locale(*lang*)

Set requests language and the matching locale.

Parameters

lang (*str*) – language code to set API localization ("en" or "fr"). Defaults to 'fr'.

tags_to_dict(*tags=<class 'dict'>*, *prev_query=<class 'dict'>*, *duplicated='rename'*)

Reverse search tags dictionary to values as keys. Useful to populate filters comboboxes for example.

Parameters

- **tags** (*dict*) – tags dictionary from a search request
- **prev_query** (*dict*) – query parameters returned after a search request. Typically *search.get("query")*.
- **duplicated** (*str*) – what to do about duplicated tags label. Values:
  - ignore - last tag parsed survives
  - merge - add duplicated in value as separated list (sep = '||')
  - rename [default] - if duplicated tag labels are part of different workgroup, so the tag label is renamed with workgroup.

PYTHON MODULE INDEX

INDEX