Isogeo PySDK

Release 3.1.0

Aug 19, 2019

Contents:

Author Julien M. (Isogeo)

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

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

CHAPTER 1

Indices and tables

- genindex
- modindex
- search

1.1 Installation

1.1.1 Get it from pip

```
pip install isogeo-pysdk
```

1.2 Package modules

1.2.1 isogeo_pysdk

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_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).

account
Get authenticated user account(= profile) informations.

Parameters

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

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
```

update(*account: isogeo_pysdk.models.user.User*, *caching: bool = 1*) → isogeo_pysdk.models.user.User
Update authenticated user account(= profile) informations.

Parameters

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

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.

application(*application_id: str, include: list = ['_abilities', 'groups']*) → isogeo_pysdk.models.application.Application
Get details about a specific application.

Parameters

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

associate_group(*application: isogeo_pysdk.models.application.Application, workgroup: isogeo_pysdk.models.workgroup.Workgroup, force: bool = 0*) → tuple
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 *canHaveManyGroups* property

create(*application: object = {'_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}, check_exists: int = 1*) → isogeo_pysdk.models.application.Application
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

delete(*application_id: str*)
Delete a application from Isogeo database.

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

dissociate_group(*application: isogeo_pysdk.models.application.Application, workgroup: isogeo_pysdk.models.workgroup.Workgroup*) → tuple
Removes the association between the specified group and the specified application.

Parameters

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

exists(*application_id: str*) → bool
Check if the specified application exists and is available for the authenticated user.

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

listing
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** (*list*) – additionnal subresource to include in the response.

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

update(*application: isogeo_pysdk.models.application.Application*, *caching: bool = 1*) → isogeo_pysdk.models.application.Application
Update a application owned by a workgroup.

Parameters

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

workgroups
Get all groups associated with an application.

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

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 (conditions).

associate_metadata(*metadata: isogeo_pysdk.models.metadata.Metadata*, *catalog: isogeo_pysdk.models.catalog.Catalog*) → requests.models.Response
Associate a metadata with a catalog.

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

Parameters

- **metadata** (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]>
```

create(*workgroup_id: str*, *catalog: isogeo_pysdk.models.catalog.Catalog*, *check_exists: bool = 1*) → isogeo_pysdk.models.catalog.Catalog
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: str*, *catalog_id: str*)
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: isogeo_pysdk.models.metadata.Metadata*, *catalog: isogeo_pysdk.models.catalog.Catalog*) → requests.models.Response
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

exists(*workgroup_id: str*, *catalog_id: str*) → bool
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

get
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

listing
Get workgroup catalogs.

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

metadata
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
Returns shares for the specified catalog.

Parameters **catalog_id** (*str*) – catalog UUID

Returns list of Shares containing the catalog

Return type list

statistics
Returns statistics for the specified catalog.

Parameters **catalog_id** (*str*) – catalog UUID

statistics_by_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

update(*catalog: isogeo_pysdk.models.catalog.Catalog*, *caching: bool = 1*) → isogeo_pysdk.models.catalog.Catalog
Update a catalog owned by a workgroup.

Parameters

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

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 (conditions).

associate_metadata(*metadata: isogeo_pysdk.models.metadata.Metadata*, *contact: isogeo_pysdk.models.contact.Contact*, *role: str = 'pointOfContact'*) → requests.models.Response
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

create(*workgroup_id: str*, *contact: isogeo_pysdk.models.contact.Contact*, *check_exists: int = 1*) → isogeo_pysdk.models.contact.Contact
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 False if a similar cataog already exists or a tuple with response error code

Return type *Contact*

delete(*workgroup_id: str*, *contact_id: str*)
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: isogeo_pysdk.models.metadata.Metadata*, *contact: isogeo_pysdk.models.contact.Contact*) → requests.models.Response
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

exists(*contact_id: str*) → bool
Check if the specified contact exists and is available for the authenticated user.

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

get(*contact_id: str*) → isogeo_pysdk.models.contact.Contact
Get details about a specific contact.

Parameters **contact_id** (*str*) – contact UUID

listing(*workgroup_id: str = None, include: list = ['count'], caching: bool = 1*) → list
Get workgroup contacts.

Parameters

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

update(*contact: isogeo_pysdk.models.contact.Contact*, *caching: bool = 1*) → isogeo_pysdk.models.contact.Contact
Update a contact owned by a workgroup.

Parameters

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

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: isogeo_pysdk.models.metadata.Metadata*, *coordinate_system: isogeo_pysdk.models.coordinates_system.CoordinateSystem*) → requests.models.Response
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: isogeo_pysdk.models.coordinates_system.CoordinateSystem*, *workgroup: isogeo_pysdk.models.workgroup.Workgroup = None*) → isogeo_pysdk.models.coordinates_system.CoordinateSystem
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.coordinate_system("4326")
```

(continues on next page)

(continued from previous page)

```
>>> # 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
    )
```

coordinate_system(*coordinate_system_code: str*, *workgroup_id: str = None*) → isogeo_pysdk.models.coordinates_system.CoordinateSystem

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.coordinate_system(srs[0].get("code")))
{
    '_tag': 'coordinate-system:4143',
    'code': 4143,
    'name': 'Abidjan 1987'
}
```

dissociate_metadata(*metadata: isogeo_pysdk.models.metadata.Metadata*) → requests.models.Response

Removes the coordinate-system from a metadata.

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

dissociate_workgroup(*coordinate_system_code: str*, *workgroup_id: str = None*) → isogeo_pysdk.models.coordinates_system.CoordinateSystem

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"
    )
```

listing (*workgroup_id: str = None*, *caching: bool = 1*) → list
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: str*, *datasource: object = {'_created': None, '_id': None, '_modified': None, '_tag': None, 'enabled': None, 'lastSession': None, 'location': None, 'name': None, 'resourceCount': None, 'sessions': None}*, *check_exists: int = 2*) → isogeo_pysdk.models.datasource.Datasource
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

datasource(*workgroup_id: str*, *datasource_id: str*) → isogeo_pysdk.models.datasource.Datasource
Get details about a specific datasource.

Parameters

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

datasources(*workgroup_id: str = None*, *include: list = []*, *caching: bool = 1*) → list
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

delete(*workgroup_id: str*, *datasource_id: str*)
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: str*, *datasource_id: str*) → bool
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

update(*workgroup_id: str*, *datasource: isogeo_pysdk.models.datasource.Datasource*, *caching: bool = 1*) → isogeo_pysdk.models.datasource.Datasource
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

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: bool = 1*) → list
Get directives.

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

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: isogeo_pysdk.models.metadata.Metadata*, *event: isogeo_pysdk.models.event.Event*) → isogeo_pysdk.models.event.Event
Add a new event to a metadata (= resource).

Parameters

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

delete(*event: isogeo_pysdk.models.event.Event*, *metadata: isogeo_pysdk.models.metadata.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: str*, *event_id: str*) → isogeo_pysdk.models.event.Event
Get details about a specific event.

Parameters

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

listing(*metadata: isogeo_pysdk.models.metadata.Metadata*) → list
Get all events of a metadata.

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

update(*event: isogeo_pysdk.models.event.Event*, *metadata: isogeo_pysdk.models.metadata.Metadata = None*) → isogeo_pysdk.models.event.Event
Update an event.

Parameters

- **event** (*class*) – Event model object to update
- **metadata** (Metadata) – parent metadata (resource) containing the 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: isogeo_pysdk.models.metadata.Metadata*, *attribute: isogeo_pysdk.models.feature_attributes.FeatureAttribute*) → isogeo_pysdk.models.feature_attributes.FeatureAttribute
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: isogeo_pysdk.models.feature_attributes.FeatureAttribute*, *metadata: isogeo_pysdk.models.metadata.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: str*, *attribute_id: str*) → isogeo_pysdk.models.feature_attributes.FeatureAttribute

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")
    )
```

import_from_dataset(*metadata_source: isogeo_pysdk.models.metadata.Metadata*, *metadata_dest: isogeo_pysdk.models.metadata.Metadata*, *mode: str = 'add'*) → bool

Import feature-attributes from another vector 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

Raises

- **TypeError** – if one metadata is not a vector
- **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)

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

listing(*metadata: isogeo_pysdk.models.metadata.Metadata*) → list

Get all feature-attributes of a metadata.

Parameters **metadata** (Metadata) – metadata (resource)

update(*attribute: isogeo_pysdk.models.feature_attributes.FeatureAttribute*, *metadata: isogeo_pysdk.models.metadata.Metadata = None*) → isogeo_pysdk.models.feature_attributes.FeatureAttribute

Update a feature-attribute.

Parameters

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

isogeo_pysdk.api.routes_format module

Isogeo API v1 - API Routes for Formats entities

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

NOTE TO DEV: *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: isogeo_pysdk.models.format.Format*, *check_exists: bool = 1*) → isogeo_pysdk.models.format.Format
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: isogeo_pysdk.models.format.Format*)
Delete a format from Isogeo database.

Parameters **frmt** (Format) – Format model object to delete

get (*format_code: str*) → isogeo_pysdk.models.format.Format
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: str = None*, *caching: bool = 1*) → list
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: str = None*, *page_size: int = 10*, *offset: int = 0*) → list
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,
```

(continues on next page)

(continued from previous page)

```
            None
        ]
    }
]
```

update(*frmt: isogeo_pysdk.models.format.Format*) → isogeo_pysdk.models.format.Format
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_pgis)
```

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 (conditions).

accept(*invitation: object = <class 'isogeo_pysdk.models.invitation.Invitation'>*) → isogeo_pysdk.models.invitation.Invitation
Accept the invitation to join an Isogeo Workgroup.

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

create(*workgroup_id: str, invitation: object = {'_created': None, '_id': None, '_modified': None, 'email': None, 'expiresIn': None, 'group': None, 'role': None}*) → isogeo_pysdk.models.invitation.Invitation
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: object = <class 'isogeo_pysdk.models.invitation.Invitation'>*) → isogeo_pysdk.models.invitation.Invitation
Decline the invitation to join an Isogeo Workgroup.

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

delete (*invitation_id: str*)
Delete an invitation from Isogeo database.

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

get (*invitation_id: str*) → isogeo_pysdk.models.invitation.Invitation
Get details about a specific invitation.

Parameters **invitation_id** (*str*) – invitation UUID

listing (*workgroup_id: str*) → list
Returns pending invitations (including expired) for the specified workgroup.

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

update (*invitation: isogeo_pysdk.models.invitation.Invitation*) → isogeo_pysdk.models.invitation.Invitation
Update a invitation owned by a invitation.

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

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 (conditions).

create (*keyword: isogeo_pysdk.models.keyword.Keyword*) → isogeo_pysdk.models.keyword.Keyword
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

delete (*keyword: isogeo_pysdk.models.keyword.Keyword*)
Delete a keyword from Isogeo database.

Parameters **keyword** (Keyword) – Keyword model object to create

get (*keyword_id: str, include: list = ['_abilities', 'count', 'thesaurus']*) → isogeo_pysdk.models.keyword.Keyword
Get details about a specific keyword.

Parameters

- **keyword_id** (*str*) – keyword UUID
- **include** (*list*) – 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
```

(continues on next page)

(continued from previous page)

```
    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)
```

metadata (*metadata_id: str = None, include: list = ['_abilities', 'count', 'thesaurus']*) → list

List a metadata's keywords with complete information.

Parameters

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

Return type list

tagging (*metadata: isogeo_pysdk.models.metadata.Metadata, keyword: isogeo_pysdk.models.keyword.Keyword, check_exists: bool = 0*) → dict

Associate a keyword to a metadata.

Parameters

- **metadata** (Metadata) – metadata (resource) to edit
- **keyword** (Keyword) – object to associate
- **check_exists** (*bool*) – check if a metadata with the same service base URL and format alerady exists. Defaults to True.

Example

```
# retrieve a metadata
md = isogeo.metadata.get(METADATA_UUID)
# retrieve a keyword
keyword = isogeo.keyword.get(KEYWORD_UUID)
```

thesaurus (*thesaurus_id: str = '1616597fbc4348c8b11ef9d59cf594c8', query: str = '', offset: int = 0, order_by: str = 'text', order_dir: str = 'desc', page_size: int = 20, specific_md: list = [], specific_tag: list = [], include: list = ['_abilities', 'count'], caching: bool = 1*) → isogeo_pysdk.models.keyword_search.KeywordSearch

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: 20.
- **specific_md** (*list*) – list of metadata UUIDs to filter on
- **specific_tag** (*list*) – list of tags UUIDs to filter on
- **include** (*list*) – subresources that should be returned. Available values:
  - '_abilities'
  - 'count'
  - 'thesaurus'

untagging(*metadata: isogeo_pysdk.models.metadata.Metadata, keyword: isogeo_pysdk.models.keyword.Keyword*) → dict

Dissociate a keyword from a metadata.

Parameters

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

workgroup(*workgroup_id: str = None, thesaurus_id: str = None, query: str = '', offset: int = 0, order_by: str = 'text', order_dir: str = 'desc', page_size: int = 20, specific_md: list = [], specific_tag: list = [], include: list = ['_abilities', 'count', 'thesaurus'], caching: bool = 1*) → isogeo_pysdk.models.keyword_search.KeywordSearch

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: 20.
- **specific_md** (*list*) – list of metadata UUIDs to filter on
- **specific_tag** (*list*) – list of tags UUIDs to filter on
- **include** (*list*) – subresources that should be returned. Available values:
  - '_abilities'
  - 'count'
  - 'thesaurus'

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: isogeo_pysdk.models.metadata.Metadata, license: isogeo_pysdk.models.license.License, description: str, force: bool = 0*) → requests.models.Response

Associate a condition (license + specific description) to a metadata. Wehn a license is associated to a metadata, it become 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."
    )
```

create(*workgroup_id: str, check_exists: int = 1, license: object = {'_abilities': None, '_id': None, '_tag': None, 'content': None, 'count': None, 'link': None, 'name': None, 'owner': None}*) → isogeo_pysdk.models.license.License

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

delete(*workgroup_id: str*, *license_id: str*)
Delete a license from Isogeo database.

Parameters

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

dissociate_metadata(*metadata: isogeo_pysdk.models.metadata.Metadata*, *condition_id: str*) → requests.models.Response
Removes the association between a metadata and a license.

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

Parameters

- **metadata** (Metadata) – metadata object to update
- **license** (License) – license model object to associate

exists(*license_id: str*) → bool
Check if the specified license exists and is available for the authenticated user.

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

get(*license_id: str*) → isogeo_pysdk.models.license.License
Get details about a specific license.

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

listing(*workgroup_id: str = None, include: list = ['_abilities', 'count'], caching: bool = 1*) → list
Get workgroup licenses.

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

update(*license: isogeo_pysdk.models.license.License*, *caching: bool = 1*) → isogeo_pysdk.models.license.License
Update a license owned by a workgroup.

Parameters

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

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: isogeo_pysdk.models.metadata.Metadata*, *limitation: isogeo_pysdk.models.limitation.Limitation*) → isogeo_pysdk.models.limitation.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: isogeo_pysdk.models.limitation.Limitation*, *metadata: isogeo_pysdk.models.metadata.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: str*, *limitation_id: str*) → isogeo_pysdk.models.limitation.Limitation
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")
    )
```

listing(*metadata: isogeo_pysdk.models.metadata.Metadata*) → list
Get limitations of a metadata.

Parameters **metadata** (Metadata) – metadata (resource)

update(*limitation: isogeo_pysdk.models.limitation.Limitation*, *metadata: isogeo_pysdk.models.metadata.Metadata = None*) → isogeo_pysdk.models.limitation.Limitation
Update a limitation.

Parameters

- **limitation** (Limitation) – Limitation model object to update
- **metadata** (Metadata) – parent metadata (resource) containing the 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: list*, *link_kind: str*) → list

Link available actions depend on link kind. Relationships between kinds and actions are described in the */link-kinds*
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 list

create(*metadata: isogeo_pysdk.models.metadata.Metadata*, *link: isogeo_pysdk.models.link.Link*) → isogeo_pysdk.models.link.Link
Add a new link to a metadata (= resource).

Parameters

- **metadata** (Metadata) – metadata (resource) to edit
- **link** (Link) – link object to create

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",
```

(continues on next page)

(continued from previous page)

```
    type="link",
    link=Link(_id=LINK_UUID)
    )
```

delete(*link: isogeo_pysdk.models.link.Link*, *metadata: isogeo_pysdk.models.metadata.Metadata = None*) → requests.models.Response
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: isogeo_pysdk.models.link.Link*, *encode_clean: bool = 1*) → tuple
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: str*, *link_id: str*) → isogeo_pysdk.models.link.Link
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
```

(continues on next page)

(continued from previous page)

```
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: bool = 1*) → list
Get the relation between kinds and action for links.

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

Return type list

listing(*metadata: isogeo_pysdk.models.metadata.Metadata*) → list
Get links of a metadata.

Parameters **metadata** (Metadata) – metadata (resource)

update(*link: isogeo_pysdk.models.link.Link*, *metadata: isogeo_pysdk.models.metadata.Metadata = None*) → isogeo_pysdk.models.link.Link
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.

upload_hosted(*metadata: isogeo_pysdk.models.metadata.Metadata*, *link: isogeo_pysdk.models.link.Link*, *file_to_upload: str*) → isogeo_pysdk.models.link.Link
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(
    title=my_file.name
    )
```

(continues on next page)

(continued from previous page)

```
# 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: isogeo_pysdk.models.metadata.Metadata*) → list
Returns asssociated catalogs with a metadata. Just a shortcut.

Parameters **metadata** (Metadata) – metadata object

Return type list

create(*workgroup_id: str*, *metadata: isogeo_pysdk.models.metadata.Metadata*, *check_exists: bool = 1*, *return_basic_or_complete: bool = 0*) → isogeo_pysdk.models.metadata.Metadata
Add a new metadata to a workgroup.

Parameters

- **workgroup_id** (*str*) – identifier of the owner workgroup
- **metadata** (Metadata) – Metadata model object to create
- **check_exists** (*bool*) – check if a metadata with the same title already exists into the workgroup:
  - 0 = no check
  - 1 = compare name [DEFAULT]
- **return_basic_or_complete** (*bool*) – 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 = basic (only the _id, title and attributes passed for the creation) [DEFAULT]
  - 1 = complete (make an addtionnal request)

Return type *Metadata*

delete(*metadata_id: str*) → requests.models.Response
Delete a metadata from Isogeo database.

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

download_xml(*metadata: isogeo_pysdk.models.metadata.Metadata*) → requests.models.Response
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: str*) → bool
Check if the specified resource exists and is available for the authenticated user.

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

get (*metadata_id: str*, *include: tuple = ()*) → isogeo_pysdk.models.metadata.Metadata
Get complete or partial metadata about a specific metadata (= resource).

Parameters

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

keywords (*metadata: isogeo_pysdk.models.metadata.Metadata, include: list = ['_abilities', 'count', 'thesaurus']*) → list
Returns asssociated keywords with a metadata. Just a shortcut.

Parameters

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

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

Return type list

update (*metadata: isogeo_pysdk.models.metadata.Metadata*) → isogeo_pysdk.models.metadata.Metadata
Update a metadata, but **ONLY** the root attributes, not the subresources.

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

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: isogeo_pysdk.models.metadata_search.MetadataSearch*)
Add shares list to the tags attributes in search.

Parameters **search** (MetadataSearch) – search to add shares

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

Parameters

- **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",
    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
)
```

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: str*, *service_url: str*, *service_type: str = 'guess'*, *service_format: str = None*, *service_title: str = None*, *check_exists: bool = 1*, *ignore_avaibility: bool = 0*, *http_verb: str = 'HEAD'*) → isogeo_pysdk.models.metadata.Metadata

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: isogeo_pysdk.models.metadata.Metadata*, *check_only: bool = 0*) → isogeo_pysdk.models.metadata.Metadata

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: isogeo_pysdk.models.metadata.Metadata*, *layer: isogeo_pysdk.models.service_layer.ServiceLayer*, *dataset: isogeo_pysdk.models.metadata.Metadata*) → requests.models.Response
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
    )
```

create(*metadata: isogeo_pysdk.models.metadata.Metadata*, *layer: isogeo_pysdk.models.service_layer.ServiceLayer*) → isogeo_pysdk.models.service_layer.ServiceLayer
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

delete(*layer: isogeo_pysdk.models.service_layer.ServiceLayer*, *metadata: isogeo_pysdk.models.metadata.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: isogeo_pysdk.models.metadata.Metadata*, *layer: isogeo_pysdk.models.service_layer.ServiceLayer*, *dataset: isogeo_pysdk.models.metadata.Metadata*) → requests.models.Response
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

layer(*metadata_id: str*, *layer_id: str*) → isogeo_pysdk.models.service_layer.ServiceLayer
Get details about a specific service_layer.

Parameters

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

listing(*metadata: isogeo_pysdk.models.metadata.Metadata*) → list
Get all service layers of a metadata.

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

update(*layer: isogeo_pysdk.models.service_layer.ServiceLayer*, *metadata: isogeo_pysdk.models.metadata.Metadata = None*) → isogeo_pysdk.models.service_layer.ServiceLayer
Update a service layer.

Parameters

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

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: isogeo_pysdk.models.metadata.Metadata*, *operation: isogeo_pysdk.models.service_operation.ServiceOperation*) → isogeo_pysdk.models.service_operation.ServiceOperation
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

listing(*metadata: isogeo_pysdk.models.metadata.Metadata*) → list
Get all operations of a metadata service.

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

operation(*metadata_id: str*, *operation_id: str*) → isogeo_pysdk.models.service_operation.ServiceOperation
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

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: isogeo_pysdk.models.share.Share*, *application: isogeo_pysdk.models.application.Application*) → tuple
Associate a share with an application.

Parameters

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

associate_catalog(*share: isogeo_pysdk.models.share.Share*, *catalog: isogeo_pysdk.models.catalog.Catalog*) → tuple
Associate a share with a catalog.

Parameters

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

associate_group(*share: isogeo_pysdk.models.share.Share*, *group: isogeo_pysdk.models.workgroup.Workgroup*) → requests.models.Response
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

create(*workgroup_id: str*, *share: object = {'_created': None, '_creator': (None,), '_id': None, '_modified': None, 'applications': None, 'catalogs': None, 'groups': None, 'name': None, 'rights': None, 'type': None, 'urlToken': None}*, *check_exists: int = 1*) → isogeo_pysdk.models.share.Share

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 inot the workgroup:

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

delete(*share_id: str*) → requests.models.Response

Delete a share from Isogeo database.

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

dissociate_application(*share: isogeo_pysdk.models.share.Share*, *application: isogeo_pysdk.models.application.Application*) → tuple

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

Parameters

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

dissociate_catalog(*share: isogeo_pysdk.models.share.Share*, *catalog: isogeo_pysdk.models.catalog.Catalog*) → tuple

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

Parameters

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

dissociate_group(*share: isogeo_pysdk.models.share.Share*, *group: isogeo_pysdk.models.workgroup.Workgroup*) → tuple

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

exists(*share_id: str*) → bool

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

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

listing

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

refresh_token(*share: isogeo_pysdk.models.share.Share*) → isogeo_pysdk.models.share.Share
Refresh the URL token of a share, used by Cartotheque, CSW, OpenCatalog.

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

reshare(*share: isogeo_pysdk.models.share.Share*, *reshare: bool = 1*) → isogeo_pysdk.models.share.Share
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

share(*share_id: str, include: list = ['_abilities', 'groups']*) → isogeo_pysdk.models.share.Share
Get details about a specific share.

Parameters

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

update(*share: isogeo_pysdk.models.share.Share*, *caching: bool = 1*) → isogeo_pysdk.models.share.Share
Update a share owned by a workgroup.

Parameters

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

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: isogeo_pysdk.models.metadata.Metadata*, *specification: isogeo_pysdk.models.specification.Specification*, *conformity: bool = 0*) → requests.models.Response
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.specification(my_specification_uuid)
>>> # associate them
>>> isogeo.specification.associate_metadata(
        metadata=md,
        specification=spec,
        conformity=1
    )
```

create(*workgroup_id: str*, *check_exists: int = 1*, *specification: object = {'_abilities': None, '_id': None, '_tag': None, 'count': None, 'isLocked': None, 'link': None, 'name': None, 'owner': None, 'published': None}*) → isogeo_pysdk.models.specification.Specification

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

delete(*workgroup_id: str*, *specification_id: str*) → dict

Delete a specification from Isogeo database.

Parameters

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

dissociate_metadata(*metadata: isogeo_pysdk.models.metadata.Metadata*, *specification_id: str*) → requests.models.Response

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** (Specification) – specification model object to associate

exists(*specification_id: str*) → bool

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

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

listing(*workgroup_id: str = None, include: list = ['_abilities', 'count'], caching: bool = 1*) → list

Get workgroup specifications.

Parameters

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

specification (*specification_id: str*) → isogeo_pysdk.models.specification.Specification
Get details about a specific specification.

Parameters **specification_id** (*str*) – specification UUID

update (*specification: isogeo_pysdk.models.specification.Specification*, *caching: bool = 1*) → isogeo_pysdk.models.specification.Specification
Update a specification owned by a workgroup.

Parameters

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

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 thesauruss (conditions).

thesauri (*caching: bool = 1*) → list
Get all thesauri.

thesaurus (*thesaurus_id: uuid.UUID = '1616597fbc4348c8b11ef9d59cf594c8', include: list = ['_abilities']*) → isogeo_pysdk.models.thesaurus.Thesaurus
Get a thesaurus.

Parameters

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

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 (conditions).

listing () → list
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
```

user(*user_id: str, include: list = ['_abilities']*) → isogeo_pysdk.models.user.User
Get details about a specific user.

Parameters

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

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 (conditions).

coordinate_systems
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: isogeo_pysdk.models.workgroup.Workgroup, check_exists: int = 1*) → isogeo_pysdk.models.workgroup.Workgroup
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]

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

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

exists (*workgroup_id: str*) → bool
Check if the specified workgroup exists and is available for the authenticated user.

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

get
Get details about a specific workgroup.

Parameters

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

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

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

invite (*workgroup_id: str*, *invitation: isogeo_pysdk.models.invitation.Invitation*) → dict
Invite new user to a workgroup. Just a shortcut.

Parameters

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

limits
Returns limits for the specified workgroup.

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

listing
Get workgroups.

Parameters

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

memberships
Returns memberships for the specified workgroup.

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

statistics
Returns statistics for the specified workgroup.

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

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

update(*workgroup: isogeo_pysdk.models.workgroup.Workgroup*, *caching: bool = 1*) → isogeo_pysdk.models.workgroup.Workgroup
Update a workgroup owned by a workgroup.

Parameters

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

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**
Bases: enum.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.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**
Bases: enum.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__)
False
>>> print("coordinateSystem" in CatalogStatisticsTags.__members__)
True
```

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

contact = 'contact'

coordinateSystem = 'coordinate-system'

format = 'format'

has_value = <bound method CatalogStatisticsTags.has_value of <enum 'CatalogStatisticsT

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**

Bases: enum.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**

Bases: enum.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**

Bases: enum.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**

Bases: enum.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:
```

(continues on next page)

(continued from previous page)

```
>>>     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**

Bases: enum.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**

Bases: enum.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**
Bases: enum.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**
Bases: enum.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**

Bases: enum.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**
Bases: enum.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**
Bases: enum.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'

has_value = <bound method MetadataSubresources.has_value of <enum 'MetadataSubresource

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**

Bases: enum.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.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'

has_value = <bound method MetadataTypes.has_value of <enum 'MetadataTypes'>>

rasterDataset = 'raster-dataset'

resource = 'resource'

service = 'service'

vectorDataset = 'vector-dataset'

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**

Bases: enum.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**
Bases: enum.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.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**
Bases: enum.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
```

(continues on next page)

(continued from previous page)

```
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'

has_value = <bound method WorkgroupStatisticsTags.has_value of <enum 'WorkgroupStatist

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: list = None*, *_created: str = None*, *_id: str = None*, *_modified: str = None*, *canHaveManyGroups: bool = None*, *client_id: str = None*, *client_secret: int = None*, *groups: list = None*, *kind: str = None*, *name: str = None*, *redirect_uris: list = None*, *scopes: list = None*, *staff: bool = None*, *type: str = None*, *url: str = 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',
                    '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_u

attr_map = {}

attr_types = {'_abilities': <class 'list'>, '_created': <class 'str'>, '_id': <clas

canHaveManyGroups
Gets the option of this Application.

Returns The option of this Application.

Return type bool

client_id
Gets the client_id of this Application.

Returns The client_id of this Application.

Return type str

client_secret
Gets the client_secret of this Application.

Returns The client_secret of this Application.

Return type str

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

Returns The groups of this Application. # noqa: E501

Return type *Workgroup*

kind
Gets the kind of this Application.

Returns The kind of this Application.

Return type str

name
Gets the name of this Application.

Returns The name of this Application.

Return type str

redirect_uris
Gets the redirect_uris of this Application.

Returns The redirect_uris of this Application.

Return type list

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

Returns The scopes of this Application. # noqa: E501

Return type *Workgroup*

staff
Gets the staff of this Application.

Returns The staff of this Application.

Return type bool

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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

Returns The type of this Application. # noqa: E501

Return type str

url
Gets the url of this Application.

Returns The url of this Application.

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: list = None*, *_created: str = None*, *_id: str = None*, *_modified: str = None*, *_tag: str = None*, *code: str = None*, *count: int = None*, *name: str = None*, *owner: isogeo_pysdk.models.workgroup.Workgroup = None*, *scan: bool = 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,
    '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,
```

(continues on next page)

(continued from previous page)

```
            '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': <clas

classmethod clean_attributes(*raw_object: dict*)
Renames attributes wich are incompatible with Python (hyphens...). See related issue: https://github.com/isogeo/isogeo-api-py-minsdk/issues/82

code
Gets the code of this Catalog.

Returns The code of this Catalog.

Return type str

count
Gets the count of this Catalog.

Returns The count of this Catalog.

Return type str

name
Gets the name of this Catalog.

Returns The name of this Catalog.

Return type str

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

Returns The owner of this Catalog. # noqa: E501

Return type *Workgroup*

scan
Gets the scan of this Catalog.

Returns The scan of this Catalog.

Return type bool

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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: list = None*, *_deleted: bool = None*, *_id: str = None*, *_tag: str = None*, *addressLine1: str = None*, *addressLine2: str = None*, *addressLine3: str = None*, *available: bool = None*, *city: str = None*, *count: int = None*, *countryCode: str = None*, *email: str = None*, *fax: str = None*, *hash: str = None*, *name: str = None*, *organization: str = None*, *owner: dict = None*, *phone: str = None*, *type: str = None*, *zipCode: str = None*, *created=None*, *modified=None*)

Bases: object

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

addressLine1
Gets the id of this Contact.

Returns The id of this Contact.

Return type str

addressLine2
Gets the id of this Contact.

Returns The second address line of this Contact.

Return type str

addressLine3
Gets the third address line of this Contact.

Returns The The third address line of this Contact.

Return type str

attr_crea = {'addressLine1': 'str', 'addressLine2': 'str', 'addressLine3': 'str', '

attr_map = {'fax': 'faxNumber', 'organization': 'organizationName', 'phone': 'phone

attr_types = {'_abilities': <class 'str'>, '_id': <class 'str'>, '_tag': <class 'st

available
Gets the availibility of this Contact.

Returns The availibility of this Contact.

Return type str

city
Gets the city of this Contact.

Returns The city of this Contact.

Return type str

count
Gets the id of this Contact.

Returns The id of this Contact.

Return type str

countryCode
Gets the country code of this Contact.

Returns The country code of this Contact.

Return type str

email
Gets the email of this Contact.

Returns The email of this Contact.

Return type str

fax
Gets the fax of this Contact.

Returns The fax of this Contact.

Return type str

hash
Gets the hash of this Contact.

Returns The hash of this Contact.

Return type str

name
Gets the name of this Contact.

Returns The name of this Contact.

Return type str

organization
Gets the organization of this Contact.

Returns The organization of this Contact.

Return type str

owner
Gets the owner of this Specification.

Returns The owner of this Specification.

Return type *Workgroup*

phone
Gets the phone number of this Contact.

Returns The phone number of this Contact.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

type
Gets the type of this Contact.

Returns The type of this Contact.

Return type str

zipCode
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: str = None*, *alias: str = None*, *code: str = None*, *name: str = None*)

Bases: object

CoordinateSystems.

Example

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

alias
Gets the custom alias of this CoordinateSystem in a workgroup.

Returns The alias of this CoordinateSystem in a workgroup.

Return type str

attr_crea = {}

attr_map = {}

attr_types = {'_tag': <class 'str'>, 'alias': <class 'str'>, 'code': <class 'str'>,

code
Gets the EPSG code of this CoordinateSystem.

Returns The EPSG code of this CoordinateSystem.

Return type str

name
Gets the name of this CoordinateSystem.

Returns The name of this CoordinateSystem.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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: list = None*, *_id: str = None*, *_modified: str = None*, *_tag: str = None*, *enabled: bool = None*, *lastSession: dict = None*, *location: str = None*, *name: str = None*, *resourceCount: int = None*, *sessions: list = 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'
```

(continues on next page)

(continued from previous page)

```
        }]
}
```

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

attr_map = {}

attr_types = {'_created': <class 'str'>, '_id': <class 'str'>, '_modified': <class

enabled
Gets the enabled of this Datasource.

Returns The enabled of this Datasource.

Return type str

lastSession
Gets the lastSession of this Datasource.

Returns The lastSession of this Datasource.

Return type dict

location
Gets the location (URL) of this Datasource.

Returns The location (URL) of this Datasource.

Return type str

name
Gets the name of this Datasource.

Returns The name of this Datasource.

Return type str

resourceCount
Gets the resourceCount of this Datasource.

Returns The resourceCount of this Datasource.

Return type *Workgroup*

sessions
Gets the sessions of this Datasource.

Returns The sessions of this Datasource.

Return type *Workgroup*

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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: str = None*, *description: str = None*, *name: str = 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 's

description
Gets the description of this Directive.

Returns The description of this Directive.

Return type str

name
Gets the name of this Directive.

Returns The name of this Directive.

Return type str

to_dict() → dict
Returns the model properties as a dict

to_str() → str
Returns the string representation of the model

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: str = None*, *date: str = None*, *description: str = None*, *kind: str = None*, *parent_resource: str = None*, *waitForSync: bool = 1*)

Bases: object

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

Example

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

(continues on next page)

(continued from previous page)

```
    'description': string,
    'kind': string
}
```

attr_crea = {'date': <class 'str'>, 'description': <class 'str'>, 'kind': <class 's

attr_map = {}

attr_types = {'_id': <class 'str'>, 'date': <class 'str'>, 'description': <class 's

date
Gets the date of this Event.

Returns The date of this Event.

Return type str

description
Gets the description of this Event.

Returns The description of this Event.

Return type str

kind
Gets the kind of this Event.

Returns The kind of this Event.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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: str = None*, *alias: str = None*, *dataType: str = None*, *description: str = None*, *language: str = None*, *name: str = None*, *parent_resource: str = None*)

Bases: object

FeatureAttributes are entities included as subresource into metadata.

Example

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

alias
Gets the alias of this FeatureAttribute.

Returns The alias of this FeatureAttribute.

Return type str

attr_crea = {'alias': <class 'str'>, 'dataType': <class 'str'>, 'description': <cla

attr_map = {}

attr_types = {'_id': <class 'str'>, 'alias': <class 'str'>, 'dataType': <class 'str

dataType
Gets the dataType of this FeatureAttribute.

Returns The dataType of this FeatureAttribute.

Return type str

description
Gets the description of this FeatureAttribute.

Returns The description of this FeatureAttribute.

Return type str

language
Gets the language of this FeatureAttribute.

Returns The language of this FeatureAttribute.

Return type str

name
Gets the name of this FeatureAttribute.

Returns The name of this FeatureAttribute.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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: str = None*, *_tag: str = None*, *aliases: list = None*, *code: str = None*, *name: str = None*, *type: str = None*, *versions: list = 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
    ]
}
```

aliases
Gets the aliases of this Format.

Returns The aliases of this Format.

Return type list

attr_crea = {'aliases': <class 'list'>, 'code': <class 'str'>, 'name': <class 'str'

attr_map = {}

attr_types = {'_id': <class 'str'>, '_tag': <class 'str'>, 'aliases': <class 'list'

code
Gets the code of this Format.

Returns The code of this Format.

Return type str

name
Gets the name of this Format.

Returns The name of this Format.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

type
Gets the type of this Format.

Returns The type of this Format.

Return type str

versions
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: str = None*, *_id: str = None*, *_modified: str = None*, *email: dict = None*, *expiresIn: str = None*, *group: str = None*, *role: bool = 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

email
Gets the email of this Invitation.

Returns The email of this Invitation.

Return type str

expiresIn
Gets the expiresIn of this Invitation.

Returns The expiresIn of this Invitation.

Return type int

group
Gets the group of this Invitation.

Returns The group of this Invitation.

Return type *Workgroup*

role
Gets the role of this Invitation.

Returns The role of this Invitation.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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: list = None*, *_created: str = None*, *_id: str = None*, *_modified: str = None*, *_tag: str = None*, *code: str = None*, *count: dict = None*, *description: str = None*, *thesaurus: isogeo_pysdk.models.thesaurus.Thesaurus = None*, *text: bool = None*)

Bases: object

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

Example

```
{
    '_abilities': [
        'keyword:delete',
        'keyword:restrict'
        ],
    '_created': None,
```

(continues on next page)

(continued from previous page)

```
    '_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': <clas

code
Gets the code of this Keyword.

Returns The code of this Keyword.

Return type str

count
Gets the count of this Keyword.

Returns The count of this Keyword.

Return type dict

description
Gets the description of this Keyword.

Returns The description of this Keyword.

Return type str

text
Gets the text of this Keyword.

Returns The text of this Keyword.

Return type bool

thesaurus
Gets the thesaurus of this Keyword. # noqa: E501

Returns The thesaurus of this Keyword. # noqa: E501

Return type *Thesaurus*

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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: int = None*, *offset: int = None*, *results: list = None*, *total: int = 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 'l

limit
Gets the created of this Keyword search.

Returns The created of this Keyword search.

Return type str

offset
Gets the offset of this Keyword search.

Returns The offset of this Keyword search.

Return type int

results
Gets the tag of this Keyword search.

Returns The tag of this Keyword search.

Return type str

to_dict() → dict
Returns the model properties as a dict

to_str() → str
Returns the string representation of the model

total
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: list = None*, *_id: str = None*, *_tag: str = None*, *count: int = None*, *content: str = None*, *link: str = None*, *name: str = None*, *owner: dict = 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 'st

content
Gets the content of this License.

Returns The content of this License.

Return type str

count
Gets the count of this License.

Returns The count of this License.

Return type int

link
Gets the link (URL) of this License.

Returns The link (URL) of this License.

Return type str

name
Gets the name of this License.

Returns The name of this License.

Return type str

owner
Gets the owner of this License.

Returns The owner of this License.

Return type *Workgroup*

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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: str = None*, *description: str = None*, *directive: int = None*, *restriction: str = None*, *type: str = None*, *parent_resource: str = 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

attr_map = {}

attr_types = {'_id': <class 'str'>, 'description': <class 'str'>, 'directive': <cla

description
Gets the description of this Limitation.

Returns The description of this Limitation.

Return type str

directive
Gets the directive of this Limitation.

Returns The directive of this Limitation.

Return type *Directive*

restriction
Gets the restriction of this Limitation.

Returns The restriction of this Limitation.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

type
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: str = None*, *actions: list = None*, *kind: str = None*, *link: dict = None*, *size: int = None*, *title: str = None*, *type: str = None*, *url: str = None*, *parent_resource: str = None*)

Bases: object

Links are entities included as subresource into metadata for data history title.

Example

```
{
    '_id': string (uuid),
    'actions': list,
    'kind': string,
    'parent_resource': string (uuid),
    'size': int,
    'title': string,
    'type': string,
    'url': string
}
```

actions

Gets the actions of this Link.

Returns The actions of this Link.

Return type list

attr_crea = {'actions': <class 'list'>, 'kind': <class 'str'>, 'link': <class 'dict

attr_map = {}

attr_types = {'_id': <class 'str'>, 'actions': <class 'list'>, 'kind': <class 'str'

kind

Gets the kind of this Link.

Returns The kind of this Link.

Return type str

link

Gets the associated link of this Link.

Returns The associated link of this Link.

Return type dict

size

Gets the size of the hosted data.

Returns The size of the hosted data.

Return type int

title
Gets the title of this Link.

Returns The title of this Link.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

type
Gets the type of this Link.

Returns The type of this Link.

Return type str

url
Gets the url of this Link.

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: list = None*, *_created: str = None*, *_creator: dict = None*, *_id: str = None*, *_modified: str = None*, *abstract: str = None*, *collectionContext: str = None*, *collectionMethod: str = None*, *conditions: list = None*, *contacts: list = None*, *coordinateSystem: dict = None*, *created: str = None*, *distance: float = None*, *editionProfile: str = None*, *encoding: str = None*, *envelope: dict = None*, *events: list = None*, *featureAttributes: list = None*, *features: int = None*, *format: str = None*, *formatVersion: str = None*, *geometry: str = None*, *keywords: list = None*, *language: str = None*, *layers: list = None*, *limitations: list = None*, *links: list = None*, *modified: str = None*, *name: str = None*, *operations: list = None*, *path: str = None*, *precision: str = None*, *published: str = None*, *scale: int = None*, *series: bool = None*, *serviceLayers: list = None*, *specifications: list = None*, *tags: list = None*, *title: str = None*, *topologicalConsistency: str = None*, *type: str = None*, *updateFrequency: str = None*, *validFrom: str = None*, *validTo: str = None*, *validityComment: str = None*)

Bases: object

Metadata are the main entities in Isogeo.

Example

```
{
    "_abilities": [
        "string"
    ],
    "_created": "string (date-time)",
    "_creator": {
        "_abilities": [
        "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",
```

(continues on next page)

(continued from previous page)

```
        "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"
        }
        }
    ],
    "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"
        },
```

(continues on next page)

(continued from previous page)

```
        "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": [
        {}
    ]
}
```

abstract
Gets the abstract.

Returns The abstract of this Metadata.

Return type str

attr_crea = {'abstract': <class 'str'>, 'collectionContext': <class 'str'>, 'collect

attr_map = {'coordinateSystem': 'coordinate-system', 'featureAttributes': 'feature-a

attr_types = {'_abilities': <class 'list'>, '_created': <class 'str'>, '_creator':

classmethod clean_attributes(*raw_object: dict*)
Renames attributes wich are incompatible with Python (hyphens...). See related issue: https://github.com/isogeo/isogeo-api-py-minsdk/issues/82

collectionContext
Gets the collectionContext of this Metadata.

Returns The collectionContext of this Metadata.

Return type str

collectionMethod
Gets the collection method of this Metadata.

Returns The collection method of this Metadata.

Return type str

conditions
Gets the conditions of this Metadata.

Returns The conditions of this Metadata.

Return type list

contacts
Gets the contacts of this Metadata.

Returns The contacts of this Metadata.

Return type list

coordinateSystem
Gets the coordinateSystem of this Metadata.

Returns The coordinateSystem of this Metadata.

Return type dict

created
Gets the created of this Metadata.

Returns The created of this Metadata.

Return type str

distance
Gets the distance of this Metadata.

Returns The distance of this Metadata.

Return type str

editionProfile
Gets the editionProfile of this Metadata.

Returns The editionProfile of this Metadata.

Return type str

encoding
Gets the encoding of this Metadata.

Returns The encoding of this Metadata.

Return type str

envelope
Gets the envelope of this Metadata.

Returns The envelope of this Metadata.

Return type str

events
Gets the events of this Metadata.

Returns The events of this Metadata.

Return type list

featureAttributes
Gets the featureAttributes of this Metadata.

Returns The featureAttributes of this Metadata.

Return type list

features
Gets the features of this Metadata.

Returns The features of this Metadata.

Return type int

format
Gets the format of this Metadata.

Returns The format of this Metadata.

Return type str

formatVersion
Gets the formatVersion of this Metadata.

Returns The formatVersion of this Metadata.

Return type str

geometry
Gets the geometry of this Metadata.

Returns The geometry of this Metadata.

Return type str

keywords
Gets the keywords of this Metadata.

Returns The keywords of this Metadata.

Return type str

language
Gets the language of this Metadata.

Returns The language of this Metadata.

Return type str

layers
Gets the layers of this Metadata.

Returns The layers of this Metadata.

Return type list

limitations
Gets the limitations of this Metadata.

Returns The limitations of this Metadata.

Return type str

links
Gets the links of this Metadata.

Returns The links of this Metadata.

Return type str

modified
Gets the modified of this Metadata.

Returns The modified of this Metadata.

Return type str

name
Gets the name of this Metadata.

Returns The name of this Metadata.

Return type str

operations
Gets the operations of this Metadata.

Returns The operations of this Metadata.

Return type list

path
Gets the path of this Metadata.

Returns The path of this Metadata.

Return type str

precision
Gets the precision of this Metadata.

Returns The precision of this Metadata.

Return type str

published
Gets the published of this Metadata.

Returns The published of this Metadata.

Return type str

scale
Gets the scale of this Metadata.

Returns The scale of this Metadata.

Return type str

series
Gets the series of this Metadata.

Returns The series of this Metadata.

Return type str

serviceLayers
Gets the serviceLayers of this Metadata.

Returns The serviceLayers of this Metadata.

Return type list

specifications
Gets the specifications of this Metadata.

Returns The specifications of this Metadata.

Return type str

tags
Gets the tags of this Metadata.

Returns The tags of this Metadata.

Return type str

title
Gets the title of this Metadata.

Returns The title of this Metadata.

Return type str

title_or_name(*slugged: bool = False*) → str
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() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

topologicalConsistency
Gets the topologicalConsistency of this Metadata.

Returns The topologicalConsistency of this Metadata.

Return type str

type
Gets the type of this Metadata.

Returns The type of this Metadata.

Return type str

updateFrequency
Gets the updateFrequency of this Metadata.

Returns The updateFrequency of this Metadata.

Return type str

validFrom
Gets the validFrom of this Metadata.

Returns The validFrom of this Metadata.

Return type str

validTo
Gets the validTo of this Metadata.

Returns The validTo of this Metadata.

Return type str

validityComment
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: dict = None*, *limit: int = None*, *offset: int = None*, *query: dict = None*, *results: list = None*, *tags: dict = None*, *total: int = 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': <clas

envelope
Gets the abilities of this Metadata search.

Returns The abilities of this Metadata search.

Return type dict

limit
Gets the created of this Metadata search.

Returns The created of this Metadata search.

Return type str

offset
Gets the offset of this Metadata search.

Returns The offset of this Metadata search.

Return type int

query
Gets the modified of this Metadata search.

Returns The modified of this Metadata search.

Return type dict

results
Gets the tag of this Metadata search.

Returns The tag of this Metadata search.

Return type list

tags
Gets the tags of this Metadata search.

Returns The tags of this Metadata search.

Return type dict

to_dict() → dict
Returns the model properties as a dict

to_str() → str
Returns the string representation of the model

total
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: str = None*, *dataset: dict = None*, *id: str = None*, *name: str = None*, *mimeTypes: str = None*, *titles: list = None*, *parent_resource: str = None*)

Bases: object

ServiceLayers are entities defining rules of data creation.

Example

```
{
    "_id": "string (uuid)",
    "id": "string",
    "mimeTypes": [
        "string"
    ],
    "titles": [
        {
            "lang": "string",
            "value": "string"
        }
    ]
}
```

attr_crea = {'name': <class 'str'>, 'titles': <class 'list'>}

attr_map = {'name': 'id'}

attr_types = {'_id': <class 'str'>, 'dataset': <class 'dict'>, 'mimeTypes': <class

dataset
Gets the dataset used for Isogeo filters of this ServiceLayer.

Returns The dataset of this ServiceLayer.

Return type dict

mimeTypes
Gets the mimeTypes of this ServiceLayer.

Returns The mimeTypes of this ServiceLayer.

Return type str

name
Gets the name used for Isogeo filters of this ServiceLayer.

Returns The name of this ServiceLayer.

Return type str

titles
Gets the titles of this ServiceLayer.

Returns The titles of this ServiceLayer.

Return type list

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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: str = None*, *mimeTypesIn: str = None*, *mimeTypesOut: str = None*, *name: str = None*, *url: str = None*, *verb: str = None*, *parent_resource: str = 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':

mimeTypesIn
Gets the mimeTypesIn used for Isogeo filters of this ServiceOperation.

Returns The mimeTypesIn of this ServiceOperation.

Return type dict

mimeTypesOut
Gets the mimeTypesOut of this ServiceOperation.

Returns The mimeTypesOut of this ServiceOperation.

Return type str

name
Gets the name used for Isogeo filters of this ServiceOperation.

Returns The name of this ServiceOperation.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

url
Gets the url of this ServiceOperation.

Returns The url of this ServiceOperation.

Return type list

verb
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: str = None*, *_creator: isogeo_pysdk.models.workgroup.Workgroup = None*, *_id: str = None*, *_modified: str = None*, *applications: list = None*, *catalogs: list = None*, *groups: list = None*, *name: str = None*, *rights: list = None*, *type: str = None*, *urlToken: str = None*)

Bases: object

Shares are entities used to publish catalog(s) of metadata to applications.

Example

```
{
    "_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",
        "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"
    },
    "_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)",
```

(continues on next page)

(continued from previous page)

```
                "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",
        "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)"
        }
    ]
}
```

applications

Gets the applications of this Share.

Returns The applications of this Share.

Return type str

```
attr_crea = {'name':  <class 'str'>, 'rights':  <class 'list'>, 'type':  <class 'str'>
```

```
attr_map = {}
```

```
attr_types = {'_created':  <class 'str'>, '_creator':  <class 'isogeo_pysdk.models.wor
```

catalogs
Gets the catalogs of this Share.

Returns The catalogs of this Share.

Return type str

groups
Gets the groups of this Share.

Returns The groups of this Share.

Return type list

name
Gets the name of this Share.

Returns The name of this Share.

Return type str

rights
Gets the rights of this Share.

Returns The rights of this Share.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

type
Gets the type of this Share.

Returns The type of this Share.

Return type str

urlToken
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: list = None*, *_id: str = None*, *_tag: str = None*, *count: int = None*, *link: str = None*, *name: str = None*, *owner: dict = None*, *published: str = 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 'st

count
Gets the id of this Specification.

Returns The id of this Specification.

Return type str

isLocked
Gets the isLocked status of this Specification.

Returns isLocked status of this Specification

Return type str

link
Gets the link (URL) of this Specification.

Returns The link (URL) of this Specification.

Return type str

name
Gets the id of this Specification.

Returns The id of this Specification.

Return type str

owner
Gets the owner of this Specification.

Returns The owner of this Specification.

Return type *Workgroup*

published
Gets the zip (postal) code of this Specification.

Returns The zip (postal) code of this Specification.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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: list = None*, *_id: str = None*, *code: str = None*, *name: str = 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 's

code
Gets the code of this Thesaurus.

Returns The code of this Thesaurus.

Return type str

name
Gets the name of this Thesaurus.

Returns The name of this Thesaurus.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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: list = None*, *_created: str = None*, *_id: str = None*, *_modified: str = None*, *contact: isogeo_pysdk.models.contact.Contact = None*, *language: str = None*, *mailchimp: dict = None*, *memberships: dict = None*, *staff: bool = None*, *timezone: str = 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",
        "phone": "string",
        "type": "string",
        "zipCode": "string"
    },
    "language": "string",
    "staff": "boolean",
    "timezone": "string"
}
```

attr_crea = {'language': <class 'str'>, 'mailchimp': <class 'str'>, 'staff': <class

attr_map = {}

attr_types = {'_abilities': <class 'list'>, '_created': <class 'str'>, '_id': <clas

contact

Gets the contact of this user.

Returns The contact of this user.

Return type *Contact*

language

Gets the id of this User.

Returns The id of this User.

Return type str

mailchimp
Gets the id of this User.

Returns The second address line of this User.

Return type str

staff
Staff status for the User.

Returns the staff status of the User

Return type bool

timezone
Gets the timezone of this User.

Returns The timezone of this User.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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: list = None*, *_created: str = None*, *_id: str = None*, *_modified: str = None*, *_tag: str = None*, *areKeywordsRestricted: bool = None*, *canCreateLegacyServiceLinks: bool = None*, *canCreateMetadata: bool = None*, *code: str = None*, *contact: isogeo_pysdk.models.contact.Contact = None*, *hasCswClient: bool = None*, *hasScanFme: bool = None*, *keywordsCasing: str = None*, *limits: dict = None*, *metadataLanguage: str = None*, *themeColor: str = None*)

Bases: object

Workgroups are entities containing metadata.

Example

```
{
    '_abilities': [
        'group:manage',
        'group:update'
    ],
    '_created': '2015-05-21T12:08:16.4295098+00:00',
```

(continues on next page)

(continued from previous page)


```
    '_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'
}
```


areKeywordsRestricted

Gets the areKeywordsRestricted of this Workgroup.

Returns The areKeywordsRestricted of this Workgroup.

Return type str

```
attr_crea = {'canCreateLegacyServiceLinks': <class 'bool'>, 'canCreateMetadata': <cl
```

```
attr_map = {'contact': ['contact.addressLine1', 'contact.addressLine2', 'contact.addr
```

attr_types = {'_abilities': <class 'list'>, '_created': <class 'str'>, '_id': <clas

canCreateLegacyServiceLinks
Gets the canCreateLegacyServiceLinks of this Workgroup.

Returns The canCreateLegacyServiceLinks of this Workgroup.

Return type str

canCreateMetadata
Gets the canCreateMetadata of this Workgroup.

Returns The canCreateMetadata of this Workgroup.

Return type str

code
Gets the code of this Workgroup.

Returns The code of this Workgroup.

Return type str

contact
Gets the contact of this Workgroup.

Returns The contact of this Workgroup.

Return type dict

hasCswClient
Gets the hasCswClient of this Workgroup.

Returns The hasCswClient of this Workgroup.

Return type str

keywordsCasing
Gets the keywordsCasing of this Workgroup.

Returns The keywordsCasing of this Workgroup.

Return type str

limits
Gets the limits of this Workgroup.

Returns The limits of this Workgroup.

Return type dict

metadataLanguage
Gets the metadataLanguage of this Workgroup.

Returns The metadataLanguage of this Workgroup.

Return type str

themeColor
Gets the themeColor of this Workgroup.

Returns The themeColor of this Workgroup.

Return type str

to_dict() → dict
Returns the model properties as a dict

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

to_str() → str
Returns the string representation of the model

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.

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*) → True
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: str*, *md_type: str*)
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: str = 'api.isogeo.com'*, *proxies: dict = None*) → bool
Test if an internet connection is operational. Src: https://stackoverflow.com/a/20913928/2556577.

Parameters **remote_server** (*str*) – remote server used to check

check_is_uuid(*uuid_str: str*)
Check if it's an Isogeo UUID handling specific form.

Parameters **uuid_str** (*str*) – UUID string to check

check_request_parameters(*parameters: dict = <class 'dict'>*)
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: *isogeo_pysdk.exceptions.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

class isogeo_pysdk.isogeo.**Isogeo**(*auth_mode: str = 'group'*, *client_secret: str = None*, *platform: str = 'qa'*, *proxy: dict = None*, *timeout: tuple = (15, 45)*, *lang: str = 'fr'*, *app_name: str = 'isogeo-pysdk/3.1.0'*, ***kwargs*)
Bases: requests_oauthlib.oauth2_session.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").
- **app_name** (*str*) – to custom the application name and user-agent

Returns authenticated requests Session you can use to send requests to the API.

Return type requests_oauthlib.OAuth2Session

Example

```
# 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'>}

connect(*username: str = None*, *password: str = None*)

Authenticate application with user credentials and get 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)
- **password** (*str*) – user password

classmethod guess_auth_mode()

header

isogeo_pysdk.translator module

Additionnal strings to be translated from Isogeo API.

class isogeo_pysdk.translator.**IsogeoTranslator**(*lang: str = 'FR'*)
Bases: object

Makes easier the translation of Isogeo API specific strings.

Parameters **lang** (*str*) – language code to apply. EN or FR.

tr(*subdomain: str*, *string_to_translate: str = ''*) → str
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

isogeo_pysdk.utils module

Complementary set of utils to use with Isogeo API.

class isogeo_pysdk.utils.**IsogeoUtils**(*proxies: dict = {}*)
Bases: object

Complementary set of utilitary methods and functions to make it easier using Isogeo API.

API_URLS = {'prod': 'api', 'qa': 'api.qa'}

APP_URLS = {'prod': 'https://app.isogeo.com', 'qa': 'https://qa-isogeo-app.azurewebs

CSW_URLS = {'prod': 'https://services.api.isogeo.com/', 'qa': 'http://services.api.q

MNG_URLS = {'prod': 'https://manage.isogeo.com', 'qa': 'https://qa-isogeo-manage.azu

OC_URLS = {'prod': 'https://open.isogeo.com', 'qa': 'https://qa-isogeo-open.azureweb

WEBAPPS = {'csw_getcap': {'args': ('share_id', 'share_token'), 'url': 'https://serv

convert_uuid(*in_uuid: str = <class 'str'>*, *mode: bool = 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: str = 'client_secrets.json'*) → dict
Loads API credentials from a file, JSON or INI.

Parameters **in_credentials** (*str*) – path to the credentials file. By default, look for a client_secrets.json file.

encoded_words_to_text (*in_encoded_words: str*)
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 (*md_id: str = None*, *md_type: str = None*, *owner_id: str = None*, *tab: str = 'identification'*) → str
Constructs the edition URL of a metadata.

Parameters

- **md_id** (*str*) – metadata/resource UUID
- **owner_id** (*str*) – owner UUID
- **tab** (*str*) – target tab in the web form

get_isogeo_version (*component: str = 'api'*, *prot: str = 'https'*)
Get Isogeo components versions. Authentication not required.

Parameters **component** (*str*) – which platform component. Options:

- api [default]
- db
- app

get_request_base_url (*route: str*, *prot: str = 'https'*) → str
Build the request url for the specified route.

Parameters

- **route** (*str*) – route to format
- **prot** (*str*) – https [DEFAULT] or http

get_url_base_from_url_token (*url_api_token: str = 'https://id.api.isogeo.com/oauth/token'*)
Returns the Isogeo API root URL (which is not included into credentials file) from the token URL (which is always included).

Parameters **str** (*url_api_token*) – url to Isogeo API ID token generator

get_view_url (*webapp: str = '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

pages_counter (*total: int*, *page_size: int = 100*) → int
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

register_webapp (*webapp_name: str*, *webapp_args: list*, *webapp_url: str*)
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: str = 'prod'*)
Set Isogeo base URLs according to platform.

Parameters **platform** (*str*) – platform to use. Options:

- prod [DEFAULT]
- qa
- int

share_extender (*share: dict*, *results_filtered: dict*)
Extend share model with additional informations.

Parameters

- **share** (*dict*) – share returned by API
- **results_filtered** (*dict*) – filtered search result

tags_to_dict (*tags=<class 'dict'>*, *prev_query=<class 'dict'>*, *duplicated: str = '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.

1.3 API coverage and features

1.3.1 Isogeo API coverage and features

This page is about the functional coverage between Isogeo API and the Python package.

Authentication

- [X] group application (oAuth2 Credentials Grant)
- [ ] user confidential application (oAuth2 Authorization Code Grant)
- [ ] user public application (oAuth2 Implicit Grant)
- [X] token auto refresh

Resources search (GET /resources/search)

Resources search parameters:

- [X] q (query)
- [X] ob (order by)
- [X] od (order direction)
- [X] _id (filter on specific resources id list)
- [X] _include (subresources management)
- [X] _lang (French or English with complete translation)
- [X] _limit (results length)
- [X] _offset (pagination)
- [X] box (filter on WGS84 boundging box)
- [X] geo (filter on WKT polygon)
- [X] rel (geometric operation to apply on 2 previous filters)
- [X] s share segregation

Resource details (GET /resources/{rid})

Resource detailed parameters:

- [X] id (metadata UUID)
- [X] _include (subresources management)

Others:

- [X] download resource in XML ISO-1939 version
- [ ] resource with contacts subresource included (GET /resources/{rid}/contacts)
- [ ] resource with events subresource included (GET /resources/{rid}/events)
- [ ] resource with keywords subresource included (GET /resources/{rid}/keywords)
- [ ] resource with operations subresource included (GET /resources/{rid}/operationds - only for services)

Keyword details (GET /keyword/{kid})

- [X] kid (keyword UUID)
- [X] _include (subresources management)
- [X] searches for keywords in a specific workgroup (GET /groups/{gid}/keywords/search)

These requests are not publicly available.

Thesaurus (GET /thesauri)

- [X] list of available thesauri
- [X] specific thesaurus (GET /thesauri/tid)
- [X] searches for keywords in a specific thesaurus (GET /thesauri/{tid}/keywords/search)

Shares (GET /shares)

- [X] list accessible shares
- [X] specific share (GET /shares/sid)

Licenses (GET /licenses)

- [X] list licenses of a workgroup
- [X] details on a specific license (GET /license/lid)

These requests are not publicly available.

Miscellaneous & bonus

- [X] check API version
- [X] check database version
- [X] pick between different Isogeo API platform (PROD, QA, [INT])
- [X] set protocol requests to HTTPS (default) or HTTP (only for GET requests not for authentication)
- [X] get every API label automatically translated (not only INSPIRE themes)
- [X] additional search parameter to automatically get full results without have to iterate with _limit and _offset
- [X] option (*ALL*) to quickly get every subresources through _include parameter
- [X] option (*augment*) to dynamically add shares ids to a search results tags (#6)
- [X] option (*tags_as_dict*) to get tags as key/values (#26)
- [X] method to easily download Isogeo hosted data
- [X] method to easily get application properties from shares request
- [X] method to easily get metadata edition URL on https://app.isogeo.com (handle direct tabs) - #23
- [X] method to load credentials from structured files (#25)
- [X] UUID checker and converter (hex <-> urn) to handle specific Isogeo UUID

- [X] automatic check on values passed into query parameter to the API
- [-] handle proxies setting (only for basic auth - not PAC nor NTLM)

1.4 Cookbook

1.4.1 Usage

Get application properties as attributes

```
from isogeo_pysdk import Isogeo

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

# get the token
token = isogeo.connect()

# add properties as attribute
isogeo.get_app_properties(token)

    # accessing properties
print(isogeo.app_properties)


# structure of returned dict
app = {"admin_url": str,
       "creation_date": str,
       "last_update": str,
       "name": str,
       "type": str,
       "kind": str,
       "url": str
       }
```

Add shares tags to search response and as attributes

```
from isogeo_pysdk import Isogeo

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

# get the token
token = isogeo.connect()

# set augment option on True
search = isogeo.search(token, page_size=0, whole_results=0, augment=1)

# through search tags
print(search.get("tags"))
```

(continues on next page)

(continued from previous page)

```
# through attributes
print(isogeo.shares_id)
```

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. The share UUID is required: 2 methods are proposed to retrieve it.

```
from isogeo_pysdk import Isogeo

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

# get the token
token = isogeo.connect()

## -- METHOD 1 - by shares route ---------------------------------------------
# get shares
shares = isogeo.shares(token)

# get first share id
share_id = shares[0].get("_id")

## -- METHOD 2 - by augmented search -----------------------------------------
# empty search
search = isogeo.search(token,
                       page_size=0,     # get only tags, not results
                       whole_results=0,   # do not retrieve the whole application␣
↪scope
                       augment=1    # -> this parameter is what we need
                       )

# get share id from tags splitting dict key
share_id = list(isogeo.shares_id.keys())[0].split(":")[1]

## -- ONCE SHARE ID RETRIEVED ------------------------------------------------

# make an augmented share request
share_augmented = isogeo.share(token, share_id, augment=1)

if "oc_url" in share_augmented:
    print("OpenCatalog is set: {}"
          .format(share_augmented.get("oc_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(token,
                       page_size=1,     # get only one result
                       whole_results=0    # do not retrieve the whole application␣
↪scope
                       )
md = search.get("results")[0]

# check
if md.get("_id") in share_augmented.get("mds_ids"):
    print("Metadata is present in this share")
else:
    print("No present").
```

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")

# 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
token = isogeo.connect()
```

Keys of returned dict:

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

URL Builder for web applications

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:

```
from isogeo_pysdk import IsogeoUtils
utils = IsogeoUtils()
url = utils.get_edit_url(md_id="0269803d50c446b09f5060ef7fe3e22b",
                         md_type="vector-dataset",
                         owner_id="32f7e95ec4e94ca3bc1afda960003882",
                         tab="attributes")
```

Get OpenCatalog URL for a metadata

```
from isogeo_pysdk import IsogeoUtils
utils = IsogeoUtils()
oc_url = utils.get_view_url(webapp="oc",
                            md_id="0269803d50c446b09f5060ef7fe3e22b",
                            share_id="1e07910d365449b59b6596a9b428ecd9",
                            share_token="TokenOhDearToken")
```

Get CSW GetCapabilities for a share

```
from isogeo_pysdk import IsogeoUtils
utils = IsogeoUtils()
csw_getcap_url = utils.get_view_url(webapp="csw_getcap",
                                    share_id="1e07910d365449b59b6596a9b428ecd9",
                                    share_token="TokenOhDearToken")
```

Get CSW GetRecords for a share

```
from isogeo_pysdk import IsogeoUtils
utils = IsogeoUtils()

csw_getrecords_url = utils.get_view_url(webapp="csw_getrecords",
                                        share_id="ShareUniqueIdentifier",
                                        share_token="TokenOhDearToken")
```

Get CSW GetRecordById for a metadata

```
from isogeo_pysdk import IsogeoUtils
utils = IsogeoUtils()

uuid_md_source = "82e73458e29a4edbaf8bfce9e16fa78b"

csw_getrecord_url = utils.get_view_url(webapp="csw_getrec",
                                       md_uuid_urn=utils.convert_uuid(uuid_md_source,␣
↪2),
                                       share_id="ShareUniqueIdentifier",
                                       share_token="TokenOhDearToken")
```

Register a custom webapp and get URL

```
from isogeo_pysdk import IsogeoUtils
utils = IsogeoUtils()
# 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 = utils.get_view_url(md_id="0269803d50c446b09f5060ef7fe3e22b",
                                webapp="PPIGE v3")
```

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:

```
from isogeo_pysdk import Isogeo

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

# get the token
token = isogeo.connect()

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

# loop on results and export
for md in search_to_be_exported.get("results"):
    title = md.get('title')
    xml_stream = isogeo.xml19139(token,
                                 md.get("_id")
                                 )

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

Others examples:

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

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:

```
from isogeo_pysdk import Isogeo

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

# get the token
token = isogeo.connect()

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

(continues on next page)

(continued from previous page)

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

Example:

- Batch export hosted data in the package sample.

Python Module Index

i

Index

A

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z