sbg Documentation

Release 0.0.1.dev0+local

Seven Bridges Genomics

Mar 18, 2024

CONTENTS

sevenbridges-python is a Python library that provides an interface for the Seven Bridges Platform and Cancer Genomics Cloud public APIs.

The Seven Bridges Platform is a cloud-based environment for conducting bioinformatic analyses. It is a central hub for teams to store, analyze, and jointly interpret their bioinformatic data. The Platform co-locates analysis pipelines alongside the largest genomic datasets to optimize processing, allocating storage and compute resources on demand.

The The Cancer Genomics Cloud (CGC), powered by Seven Bridges, is also a cloud-based computation environment. It was built as one of three pilot systems funded by the National Cancer Institute to explore the paradigm of colocalizing massive genomics datasets, like The Cancer Genomics Atlas (TCGA), alongside secure and scalable computational resources to analyze them. The CGC makes more than a petabyte of multi-dimensional data available immediately to authorized researchers. You can add your own data to analyze alongside TCGA using predefined analytical workflows or your own tools.

CHAPTER

ONE

CONTENT

1.1 Installation

The easiest way to install sevenbridges-python is using pip.

```
$ pip install sevenbridges-python
```

1.2 Get the Code

sevenbridges-python is actively developed on GitHub, where the code is always available.

The easiest way to obtain the source is to clone the public repository:

```
$ git clone git://github.com/sbg/sevenbridges-python.git
```

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

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

If you are interested in reviewing this documentation locally, clone the repository and invoke:

::

    $ make html

from the docs folder.

1.3 Quickstart

On this page, you'll find a reference for the Seven Bridges API Python client.

We encourage you to consult our other API resources:

- The Seven Bridges Github repository, okAPI, which includes Python example scripts such as recipes (which allow you to perform specific tasks) and tutorials (which will walk you through entire analyses) via the API. These recipes and tutorials make use of the sevenbridges-python bindings below.
- The Seven Bridges API documentation on our Knowledge Center, which includes a reference collection of API requests to help you get started right away.

1.3.1 Authentication and Configuration

In order to authenticate with the API, you should pass the following items to sevenbridges-python:

(a) Your authentication token

(b) The API endpoint you will be interacting with. This is either the endpoint for the Seven Bridges Platform or for the Seven Bridges Cancer Genomics Cloud (CGC) or for CAVATICA.

You can find your authentication token on the respective pages:

- https://igor.sbgenomics.com/developer for the Seven Bridges Platform
- https://cgc.sbgenomics.com/developer for the CGC
- https://cavatica.sbgenomics.com/developer for Cavatica

The API endpoints for each environment are:

- https://api.sbgenomics.com/v2 for the Seven Bridges Platform
- https://cgc-api.sbgenomics.com/v2 for the CGC.
- https://cavatica-api.sbgenomics.com/v2 for CAVATICA

Note: We will see below how to supply information about your auth token and endpoint to the library.

For more information about the API, including details of the available parameters for each API call, you should check the API documentation before using this library:

- http://docs.sevenbridges.com/docs/the-api for the Seven Bridges Platform.
- http://docs.cancergenomicscloud.org/docs/the-cgc-api for the CGC.
- http://docs.cavatica.org/docs/the-api for CAVATICA

1.3.2 How to use the Quickstart

We recommend that you pay particular attention to the section 'Managing Projects' of this Quickstart, since it contains general information on working with any kind of Platform or CGC resource (projects, files, tasks, etc) via the Python methods available in this library.

1.3.3 Initializing the library

Once you have obtained your authentication token from one of the URLs listed above, you can initialize the `Api` object defined by this library by passing in your authentication token and endpoint. There are three methods to do this. Details of each method are given below:

1. Pass the parameters `url` and `token` and optional `proxies` explicitly when initializing the API object.
2. Set the API endpoint and token to the environment variables `SB_API_ENDPOINT` and `SB_AUTH_TOKEN` respectively.
3. Use a configuration file `$HOME/.sevenbridges/credentials` with the defined credentials parameters. If config is used proxy settings will be read from `$HOME/.sevenbridges/sevenbridges-python/config` .ini like file for section `[proxies]`

Note: Keep your authentication token safe! It encodes all your credentials on the Platform or CGC. Generally, we recommend storing the token in a configuration file, which will then be stored in your home folder rather than in the code itself. This prevents the authentication token from being committed to source code repositories.

Import the library

You should begin by importing the API library sevenbridges-python to your python script that will interact with the API:

```
import sevenbridges as sbg
```

Then, use one of the following three methods to initialize the library:

1. Initialize the library explicitly

The library can be also instantiated explicitly by passing the URL and authentication token as key-value arguments into the `Api` object.

```
api = sbg.Api(url='https://api.sbgenomics.com/v2', token='<TOKEN_HERE>')
```

Note - you can initialize several API clients with different credentials or environments.

2. Initialize the library using environment variables

```
import os

# Usually these variables would be set in the shell beforehand
os.environ['SB_API_ENDPOINT'] = '<https://api.sbgenomics.com/v2' # or 'https://cgc-api.
↪sbgenomics.com/v2>' for cgc, or 'https://cavatica-api.sbgenomics.com/v2' for cavatica
os.environ['SB_AUTH_TOKEN'] = '<TOKEN_HERE>'

api = sbg.Api()
```

3. Initialize the library using a configuration file

The configuration file, `$HOME/.sevenbridges/credentials`, has a simple `.ini` file format, with the environment (the Seven Bridges Platform, or the CGC, or Cavatica) indicated in square brackets, as shown:

```
[default]
api_endpoint = https://api.sbgenomics.com/v2
auth_token = <TOKEN_HERE>

[cgc]
api_endpoint = https://cgc-api.sbgenomics.com/v2
auth_token = <TOKEN_HERE>

[cavatica]
```

(continues on next page)

(continued from previous page)

```
api_endpoint = https://cavatica-api.sbgenomics.com/v2
auth_token = <TOKEN_HERE>
```

The `Api` object is the central resource for querying, saving and performing other actions on your resources on the Seven Bridges Platform or CGC. Once you have instantiated the configuration class, pass it to the API class constructor.

```
c = sbg.Config(profile='cgc')
api = sbg.Api(config=c)
```

If not profile is set it will use the default profile.

Note: if user creates the api object `api=sbg.Api()` and does not pass any information the library will first search whether the environment variables are set. If not it will check if the configuration file is present and read the `[default]` profile. If that also fail it will raise an exception

---

Advance Access Features

Advance access features are subject to a change. To enable them just pass the `advance_access=True` flag when instantiating the library

```
api = sbg.Api(url='https://api.sbgenomics.com/v2', token='<TOKEN_HERE>', advance_
↪access=True)
```

Note:

- Advance access features are subject to a change. No guarantee of any sort is given for AA API calls maintainability.

If you fully understand the above mentioned limitation of Advance access features and are certain you want to use the features across your scripts, you can set this in the *$HOME/.sevenbridges/sevenbridges-python/config* configuration file.

```
[mode]
advance_access=True
```

1.3.4 Proxy configuration

Proxy configuration can be supplied in three different ways.

- explicit initialization

```
api = sb.Api(url='https://api.sbgenomics.com/v2', token='<TOKEN_HERE>',
       proxies={'https_proxy':'host:port', 'http_proxy': 'host:port'})
```

- environment variables

```
os.environ['HTTP_PROXY'] = 'host:port'
os.environ['HTTPS_PROXY'] = 'host:port'
```

- *$HOME/.sevenbridges/sevenbridges-python/config* configuration file

```
[proxies]
https_proxy=host:port
http_proxy=host:port
```

- Explicit with config

```
config = sb.Config(profile='my-profile',
                   proxies={'https_proxy':'host:port', 'http_proxy': 'host:port
↪'})
api = sb.Api(config=config)
```

Note: Once you set the proxy, all calls including upload and download will use the proxy settings.

1.3.5 Low level configuration

For low level library tweaking such as: number of cached connections and number of parallel requests (useful only for multi-thread applications) there are few useful arguments when instantiating Api.

- Three arguments are directly exposed from requests library. Argument *pool_connections* is the number of urllib3 connection pools to cache. Argument *pool_maxsize* gives ability to constraint maximum number of parallel requests to save in the pool while *pool_block* flag tells whether the connection pool should block for connections. More detailed explanation of those arguments can be found here.

```
api = sb.Api(
    pool_connections=<CONN_NUMBER>,
    pool_maxsize=<POOL_SIZE>,
    pool_block=<BLOCK_FLAG>
)
```

- There is a way to throttle number of parallel requests over all connection pools. That is done using *max_parallel_requests* argument.

```
api = sb.Api(max_parallel_requests=<MAX_PARALLEL>)
```

Note: Changing those values from default could affect performance.

1.3.6 Rate limit

For API requests that require authentication (i.e. all requests, except the call to list possible API paths), you can issue a maximum of 1000 requests per 300 seconds. Note that this limit is generally subject to change, depending on API usage and technical limits. Your current rate limit, the number of remaining requests available within the limit, and the time until your limit is reset can be obtained using your `Api` object, as follows.

```
api.limit
api.remaining
api.reset_time
```

1.3.7 Error Handlers

Error handler is a callable that accepts the `api` and `response` objects and returns the response object. They are most useful when additional logic needs to be implemented based on request result.

Example:

```
def error_handler(api, response):
    # Do something with the response object
    return response
```

sevenbridges-python library comes bundled with several useful error handlers. The most used ones are `maintenance_sleeper` and `rate_limit_sleeper` which pause your code execution until the SevenBridges/CGC public API is in maintenance mode or when the rate limit is breached.

Usage:

```
from sevenbridges.http.error_handlers import rate_limit_sleeper, maintenance_sleeper
api = sb.Api(url='https://api.sbgenomics.com/v2', token='<TOKEN_HERE>',
        error_handlers=[rate_limit_sleeper, maintenance_sleeper])
```

Note: Api object instantiated in this way with error handlers attached will be resilient to server maintenance and rate limiting.

1.3.8 Resource

Most objects handled by sevenbridges-python are inherited from the `Resource` model, which contains some basic methods for all inherited resources.

Resources support lazy fetching, so if a Resource is created with an `id` or `href` the client will fetch it from the API when any attribute is accessed directly (with a dot operator).

All resources contain the `api` property which is generally set by the client to be the previously configured default one, but it can be overridden on a Resource level.

All resources implement the following class methods:

- `get(id='resource_id')` - Sends a GET request to the API to retrieve the resource

All resources implement the following instance methods:

- `reload()` - Re-initializes the resource with it's data from the API server
- `delete()` - Sends a DELETE request to the API to delete the resource
- `field(name='field_name')` - Reads a field value without lazy fetching

1.3.9 Managing users

Currently any authenticated user can access his or her information by invoking the following method:

```
me = api.users.me()
```

Once you have initialized the library by authenticating yourself, the object `me` will contain your user information. This includes:

```
me.href
me.username
me.email
me.first_name
me.last_name
me.affiliation
me.phone
me.address
me.city
me.state
me.zip_code
me.country
me.role
```

For example, to obtain your email address invoke:

```
me.email
```

For user management in divisions and teams, the following are available:

```
# All users in division
users = api.users.query(division='<division_slug>')

# Division members
members = api.users.query(division='<division_slug>', role='member')

# Division admins
admins = api.users.query(division='<division_slug>', role='admin')

# Division external collaborators
external = api.users.query(division='<division_slug>', role='external_collaborator')

# Teams for a division
teams = api.teams.query(division='<division_slug>')

# All teams in division
teams_all = api.teams.query(division='<division_slug>', list_all=True)
```

For disabling users:

```
user = api.users.get('<username>')
user.disable()
```

1.3.10 Managing projects

There are several methods on the `Api` object that can help you manage your projects.

Note: If you are not familiar with the project structure of the Seven Bridges Platform and CGC, take a look at their respective documentation: projects on the CGC and projects on the Seven Bridges Platform.

List Projects - introduction to pagination and iteration

In order to list your projects, invoke the `api.projects.query` method. This method follows server pagination and therefore allows pagination parameters to be passed to it. Passing a pagination parameter controls which resources you are shown. The `offset` parameter controls the start of the pagination while the `limit` parameter controls the number of items to be retrieved.

Note: See the Seven Bridges API overview or the CGC API overview for details of how to refer to a project, and for examples of the pagination parameters.

Below is an example of how to get all your projects, using the `query` method and the pagination parameters `offset` of 0 and `limit` of 10.

```
project_list = api.projects.query(offset=0, limit=10)
```

`project_list` has now been defined to be an object of the type **collection** which acts just like a regular python list, and so supports indexing, slicing, iterating and other list functions. All collections in the sevenbridges-python library have two methods: `next_page` and `previous_page` which allow you to load the next or previous pagination pages.

There are several things you can do with a **collection** of any kind of object:

1. The generic query, e.g. `api.projects.query()`, accepts the pagination parameters `offset` and `limit` as introduced above.
2. If you wish to iterate on a complete **collection** use the `all()` method, which returns an iterator
3. If you want to manually iterate on the **collection** (page by page), use `next_page()` and `previous_page()` methods on the collection.
4. You can easily cast the **collection** to the list, so you can re-use it later by issuing the standard Python `project_list = list(api.projects.query().all())`.

```
# Get details of my first 10 projects.
project_list = api.projects.query(limit=10)
```

```
# Query projects by project name
project_list = api.projects.query(name=project_name)
```

```
# Query projects by project category
project_list = api.projects.query(category=project_category)
```

```
# List projects that have both tags 'tagA' and 'tagB'
project_list = api.projects.query(tags=['tagA', 'tagB'])
```

```
# Iterate through all my projects and print their name and id
for project in api.projects.query().all():
    print (project.id,project.name)
```

```
# Get all my current projects and store them in a list
my_projects = list(api.projects.query().all())
```

Get details of a single project

You can get details of a single project by issuing the `api.projects.get()` method with the parameter `id` set to the id of the project in question. Note that this call, as well as other calls to the API server may raise an exception which you can catch and process if required.

Note - To process errors from the library, import `SbgError` from `sevenbridges.errors`, as shown below.

```
from sevenbridges.errors import SbgError
try:
    project_id = 'doesnotexist/forsure'
    project = api.projects.get(id=project_id)
except SbgError as e:
    print (e.message)
```

Errors in `SbgError` have the properties `code` and `message` which refer to the number and text of 4-digit API status codes that are specific to the Seven Bridges Platform and API. To see all the available codes, see the documentation:

- http://docs.sevenbridges.com/docs/api-status-codes for the Seven Bridges Platform
- http://docs.cancergenomicscloud.org/docs/api-status-codes for the CGC.

Project properties

Once you have obtained the `id` of a Project instance, you can see its properties. All projects have the following properties:

`href` - Project href on the API

`id` - Id of the project

`name` - name of the project

`description` - description of the project

`billing_group` - billing group attached to the project

`type` - type of the project (v1 or v2)

`tags` - list of project tags

`root_folder` - Id of project's root folder

`settings` - Project settings

`created_by` - Project creator

`created_on` - Date of creation

`modified_on` - Modification date

The property href `href` is a URL on the server that uniquely identifies the resource in question. All resources have this attribute. Each project also has a name, identifier, description indicating its use, a type, some tags and also a billing_group identifier representing the billing group that is attached to the project.

Project methods – an introduction to methods in the sevenbridges-python library

There are two types of methods in the sevenbridges-python library: static and dynamic. Static methods are invoked on the `Api` object instance. Dynamic methods are invoked from the instance of the object representing the resource (e.g. the project).

Static methods include:

1. Create a new resource: for example, `api.projects.create(name="My new project", billing_group='296a98a9-424c-43f3-aec5-306e0e41c799')` creates a new resource. The parameters used will depend on the resource in question.
2. Get a resource: the method `api.projects.get(id='user/project')` returns details of a specific resource, denoted by its id.
3. Query resources - the method `api.projects.query()` method returns a pageable list of type `collection` of projects. The same goes for other resources, so `api.tasks.query(status='COMPLETED')` returns a **collection** of completed tasks with default paging.

Dynamic methods can be generic (for all resources) or specific to a single resource. They are called on a concrete object, such as a `Project` object.

So, suppose that `project` is an instance of `Project` object. Then, we can:

1. Delete the resource: `project.delete()` deletes the object (if deletion of this resource is supported on the API).
2. Reload the resource from server: `project.reload()` reloads the state of the object from the server.
3. Save changes to the server: `project.save()` saves all properties

The following example shows some of the methods used to manipulate projects.

```
# Get a collection of projects
projects = api.projects.query()

# Grab the first billing group
bg = api.billing_groups.query(limit=1)[0]

# Create a project using the billing group grabbed above
new_project = api.projects.create(name="My new project", billing_group=bg.id)

# Add a new member to the project
new_project.add_member(user='newuser', permissions= {'write':True, 'execute':True})
```

Other project methods include:

1. Get members of the project and their permissions - `project.get_members()` - returns a `Collection` of members and their permissions
2. Add a member to the project - `project.add_member()`
3. Add a team member to the project - `project.add_member_team()`
4. Add a division member to the project - `project.add_member_division()`
5. Remove a member from the project - `project.remove_member()`

6. List files from the project - `project.get_files()`
7. Add files to the project - `project.add_files()` - you can add a single `File` or a `Collection` of files
8. List apps from the project - `project.get_apps()`
9. List tasks from the project - `project.get_tasks()`

1.3.11 Managing datasets

The Cavatica Datasets API functionality is an advance access feature which allows you to manage datasets and their members using dedicated API calls.

The following operations are supported:

- `query()` - Query all datasets
- `get_owned_by()` - Get all datasets owned by a provided user
- `get()` - Get dataset with the provided id
- `save()` - Save changes to a dataset
- `delete()` - Delete dataset
- `get_members()` - Get all members of a dataset
- `get_member()` - Get details on a member of a dataset
- `add_member()` - Add a member to a dataset
- `remove_member()` - Remove member from a dataset

Dataset properties

Each dataset has the following properties:

`href` - The URL of the dataset on the API server.

`id` - Dataset identifier.

`name` - Dataset name.

`description` - Dataset description.

Member permissions

Dataset permissions can be accessed and edited directly on the member object.

Examples

```
# List all public datasets
datasets = api.datasets.query(visibility='public')

# List all datasets owned by user
datasets = api.datasets.query()

# List datasets by owner
```

(continues on next page)

(continued from previous page)

```
datasets_by_owner = api.datasets.get_owned_by('dataset_owner')

# Get details of a dataset
dataset = api.datasets.get('dataset_owner/dataset-name')

# List members of a dataset
members = dataset.get_members()

# Get details of a dataset member
member = dataset.get_member('dataset_member')

# Modify a dataset member's permissions
member.permissions['execute'] = False
member.save()

# Get a dataset member's permissions
permissions = member.permissions

# List dataset files
files = api.files.query(dataset=dataset)

# Edit a dataset
dataset.description = 'A new description'
dataset.save()

# Remove a dataset member
dataset.remove_member(member=member)

# Add a dataset member
added_member = dataset.add_member(
    username='new_member',
    permissions={
        "write": True,
        "read": True,
        "copy": False,
        "execute": True,
        "admin": False
    }
)

# Delete a dataset
dataset.delete()
```

1.3.12 Manage billing

There are several methods on the `Api` object to can help you manage your billing information. The billing resources that you can interact with are *billing groups* and *invoices*.

Manage billing groups

Querying billing groups will return a standard **collection** object.

```
# Query billing groups
bgroup_list = api.billing_groups.query(offset=0, limit=10)
```

```
# Fetch a billing group's information
bg = api.billing_groups.get(id='f1969c90-da54-4118-8e96-c3f0b49a163d')
```

Billing group properties

The following properties are attached to each billing group:

`href` - Billing group href on the API server.

`id` - Billing group identifier.

`owner` - Username of the user that owns the billing group.

`name` - Billing group name.

`type` - Billing group type (free or regular)

`pending` - True if billing group is not yet approved, False if the billing group has been approved.

`disabled` - True if billing group is disabled, False if its enabled.

`balance` - Billing group balance.

Billing group methods

Billing group methods:

`analysis_breakdown()` fetches analysis breakdown for the selected billing group.

`storage_breakdown()` fetches storage breakdown for the selected billing group.

`egress_breakdown()` fetches egress breakdown for the selected billing group.

Manage invoices

Querying invoices will return an Invoices **collection** object.

```
invoices = api.invoices.query()
```

Once you have obtained the invoice identifier you can also fetch specific invoice information.

```
invoices = api.invoices.get(id='6351830069')
```

Invoice properties

The following properties are attached to each invoice.

`href` - Invoice href on the API server.

`id` - Invoice identifier.

`pending` - Set to `True` if invoice has not yet been approved by Seven Bridges, `False` otherwise.

`analysis_costs` - Costs of your analysis.

`storage_costs` - Storage costs.

`total` - Total costs.

`invoice_period` - Invoicing period (from-to)

1.3.13 Managing files and folders

Files are an integral part of each analysis. As for as all other resources, the sevenbridges-python library enables you to effectively query files, in order to retrieve each file's details and metadata. The request to get a file's information can be made in the same manner as for projects and billing, presented above.

Folders are represented as files with the type "folder".

The available methods for fetching specific files are `query` and `get`:

```
# Query all files in a project
file_list = api.files.query(project='user/my-project')
```

```
# Get a single file's information
file = api.files.get(id='5710141760b2b14e3cc146af')
```

File properties

Each file has the following properties:

`href` - File href on the API server.

`id` - File identifier.

`type` - File type.

`name` - File name.

`size` - File size in bytes.

`parent` - Parent folder.

`project` - Identifier of the project that file is located in.

`created_on` - Date of the file creation.

`modified_on` - Last modification of the file.

`origin` - File origin information, indicating the task that created the file.

`tags` - File tags.

`metadata` - File metadata.

File methods

Files have the following methods:

- Refresh the file with data from the server: `reload()`
- Copy the file from one project to another: `copy()`
- Download the file: `download()`
- Save modifications to the file to the server `save()`
- Delete the resource: `delete()`
- List files in folder: `list_files()`
- Create folder: `create_folder()`
- Copy file to folder: `copy_to_folder()`
- Move file to folder: `move_to_folder()`

See the examples below for information on the arguments these methods take:

Examples

```
# Filter files by name to find only file names containing the specified string:
files = api.files.query(project='user/my-project')
my_file = [file for file in files if 'fasta' in file.name]

# Or simply query files by name if you know their exact file name(s)
files = api.files.query(project='user/myproject', names=['SRR062634.filt.fastq.gz',
↪'SRR062635.filt.fastq.gz'])
my_files = api.files.query(project='user/myproject', metadata = {'sample_id': 'SRR062634
↪'} )

# Query contents of a folder
parent = api.files.get('parent_folder_id')
result = api.files.query(parent=parent)

# Edit a file's metadata
my_file = my_files[0]
my_file.metadata['sample_id'] = 'my-sample'
my_file.metadata['library'] = 'my-library'


# Add metadata (if you are starting with a file without metadata)
my_file = my_files[0]
my_file.metadata = {'sample_id' : 'my-sample',
                    'library' : 'my-library'
                   }

# Also set a tag on that file
my_file.tags = ['example']

# Save modifications
my_file.save()
```

(continues on next page)

(continued from previous page)

```
# Copy a file between projects
new_file = my_file.copy(project='user/my-other-project', name='my-new-file')

# Download a file to the current working directory
# Optionally, path can contain a full path on local filesystem
new_file.download(path='my_new_file_on_disk')

# Get a folder
folder = api.files.get('file-identifier')

# List files in a folder
file_list = folder.list_files()

# Create folder (with a project or parent identifier)
new_folder = api.files.create_folder(
    name='new_folder_name', project='project-identifier',
)
new_folder = api.files.create_folder(
    name='new_folder_name', parent='parent-folder-identifier'
)

# Copy file to folder
copied_file = my_file.copy_to_folder(
    parent=new_folder, name='new-file-name'
)

# Move file to folder
moved_file = my_file.move_to_folder(
    parent=new_folder, name='new-file-name'
)
```

List Files - introduction to pagination

The results of files queries - `api.files.query()` and `api.files.list_files()`, can be paginated in two ways: using offset or continuation token parameter.

Working with offset pagination is equivalent to that explained in 'List Projects - introduction to pagination and iteration' section, so one can get informed there about using `offset` and `limit` parameters.

The continuation token pagination can be achieved by using `cont_token` parameter (in combination with `limit`) in query methods (`query()`/`list_files()`). This parameter is a base64 encoded value, telling server where to start next page. If one wants to use continuation token pagination, `query()`/`list_files()` method should be called with `cont_token` parameter, as in example below:

```
files_list = api.files.query(cont_token='init', limit=10)
```

The `'init'` value is special initial value that should be passed at first call of `query()`/`list_files()` method if continuation token pagination is desired.

There are certain restrictions on using this parameter:

1. The continuation token pagination is advance access feature, so `cont_token` parameter can be used only if `advance_access` is set to `True`, otherwise `SbgError` will be returned.

2. The continuation token pagination cannot be used together with metadata filtering, so if `cont_token` and `metadata` parameters are both provided to `query()`/`list_files()`, `SbgError` will be returned.
3. The continuation token pagination and offset pagination are mutually exclusive, so if there are passed both `cont_token` and `offset` parameters to `query()`/`list_files()`, `SbgError` will be returned.

Search Files using SBG query language

Files can be searched based on a query criterion written in a special query language. The query syntax is explained in the documentation. This can be achieved using `search()` method:

```
search_response = api.files.search(query='IN "user/example-project" WHERE type = "FILE"')

search_response.count # Gets the number of returned files/folders (files in this␣
↪concrete example)
search_response.cont_token # Gets continuation token that is used to fetch the next page␣
↪of data
search_response.result_set # Gets the list of resulting files/folders
```

Pagination parameters

Token-based pagination can be achieved in one of the ways:

1. By using `cont_token` and `limit` parameters in `search()` method:

```
search_response = api.files.search(query='IN "user/example-project" WHERE type = "FILE"',
↪ limit=100, cont_token=start)
```

2. With `TOKEN` and `LIMIT` parameters in the provided query:

```
search_response = api.files.search(query='IN "user/example-project" WHERE type = "FILE"␣
↪LIMIT 100 TOKEN start')
```

1.3.14 Managing file upload and download

`sevenbridges-python` library provides both synchronous and asynchronous way of uploading or downloading files.

File Download

Synchronous file download:

```
file = api.files.get('file-identifier')
file.download('/home/bar/foo/file.bam')
```

Asynchronous file download:

```
file = api.files.get('file-identifier')
download = file.download('/home/bar/foo.bam', wait=False)

download.path # Gets the target file path of the download.
download.status # Gets the status of the download.
```

(continues on next page)

(continued from previous page)

```
download.progress # Gets the progress of the download as percentage.
download.start_time # Gets the start time of the download.
download.duration # Gets the download elapsed time.

download.start() # Starts the download.
download.pause() # Pauses the download.
download.resume() # Resumes the download.
download.stop() # Stops the download.
download.wait() # Block the main loop until download completes.
```

You can register the callback or error callback function to the download handle: `download.add_callback(callback=my_callback, errorback=my_error_back)`

Registered callback method will be invoked on completion of the download. The errorback method will be invoked if error happens during download.

File Upload

Synchronous file upload:

```
# Get the project or parent folder to where we want to upload files.
project = api.projects.get('project-identifier')
api.files.upload('/home/bar/foo/file.fastq', project=project)

parent_folder = api.files.get('folder-identifier')
api.files.upload('/home/bar/foo/file.fastq', parent=parent_folder)

# Optionally we can set file name of the uploaded file.
api.files.upload('/home/bar/foo/file.fastq', project, file_name='new.fastq')
```

Asynchronous file upload:

```
upload = api.files.upload('/home/bar/foo/file.fastq', 'project-identifier', wait=False)

upload.file_name # Gets the file name of the upload.
upload.status # Gets the status of the upload.
upload.progress # Gets the progress of the upload as percentage.
upload.start_time # Gets the start time of the upload.
upload.duration # Gets the upload elapsed time.

upload.start() # Starts the upload.
upload.pause() # Pauses the upload.
upload.resume() # Resumes the upload.
upload.stop() # Stops the upload.
upload.wait() # Block the main loop until upload completes.
```

You can register the callback or error callback in the same manner as it was described for asynchronous file download.

1.3.15 Managing volumes: connecting cloud storage to the Platform

Volumes authorize the Platform to access and query objects on a specified cloud storage (Amazon Web Services or Google Cloud Storage) on your behalf. As for as all other resources, the sevenbridges-python library enables you to effectively query volumes, import files from a volume to a project or export files from a project to the volume.

The available methods for listing volumes, imports and exports are `query` and `get`, as for other objects:

```
# Query all volumes
volume_list = api.volumes.query()
# Query all imports
all_imports = api.imports.query()
# Query failed exports
failed_exports = api.exports.query(state='FAILED')
```

```
# Get a single volume's information
volume = api.volumes.get(id='user/volume')
# Get a single import's information
i = api.imports.get(id='08M4ywDZkQuJOb3L5M8mMSvzoeGezTdh')
# Get a single export's information
e = api.exports.get(id='0C7T8sBDP6aiNbwvXv12QZFPW55wJ3GJ')
```

Volume properties

Each volume has the following properties:

`href` - Volume href on the API server.

`id` - Volume identifier in format owner/name.

`name` - Volume name. Learn more about this in our Knowledge Center.

`access_mode` - Whether the volume was created as read-only (RO) or read-write (RW). Learn more about this in our Knowledge Center.

`active` - Whether or not this volume is active.

`created_on` - Time when the volume was created.

`modified_on` - Time when the volume was last modified.

`description` - An optional description of this volume.

`service` - This object contains the information about the cloud service that this volume represents.

Volume methods

Volumes have the following methods:

- Refresh the volume with data from the server: `reload()`
- Get imports for a particular volume `get_imports()`
- Get exports for a particular volume `get_exports()`
- Create a new volume based on the AWS S3 service using IAM user as authentication mechanism - `create_s3_volume()`

- Create a new volume based on the AWS S3 service using IAM role as authentication mechanism - `create_s3_volume_role_auth()`
- Create a new volume based on the Google Cloud Storage service - `create_google_volume()`
- Create a new volume based on the Aliyun service - `create_oss_volume()`
- Save modifications to the volume to the server `save()`
- Unlink the volume `delete()`
- Get volume members `get_members()`
- Add a member to the project - `add_member()`
- Add a team member to the project - `add_member_team()`
- Add a division member to the project - `add_member_division()`
- List files that belong to a volume - `list()`

See the examples below for information on the arguments these methods take:

Examples

```
# Create a new volume based on AWS S3 for importing files, with IAM user as auth␣
↪mechanism
volume_import = api.volumes.create_s3_volume(
    name='my_input_volume',
    bucket='my_bucket',
    access_key_id='AKIAIOSFODNN7EXAMPLE',
    secret_access_key = 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY',
    access_mode='RO'
)

# Create a new volume based on AWS S3 for exporting files, with IAM user as auth␣
↪mechanism
volume_export = api.volumes.create_s3_volume(
    name='my_output_volume',
    bucket='my_bucket',
    access_key_id='AKIAIOSFODNN7EXAMPLE',
    secret_access_key = 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY',
    access_mode='RW'
)

# Create a new volume based on AWS S3 for importing files, with IAM role as auth␣
↪mechanism
volume_import = api.volumes.create_s3_volume_role_auth(
    name='my_input_volume',
    bucket='my_bucket',
    role_arn='arn:aws:iam::ACCOUNTNUMBER:role/ROLE-NAME',
    external_id = 'external_id',
    access_mode='RO'
)

# Create a new volume based on AWS S3 for exporting files, with IAM role as auth␣
↪mechanism
```

(continues on next page)

(continued from previous page)

```
volume_export = api.volumes.create_s3_volume(
    name='my_output_volume',
    bucket='my_bucket',
    role_arn='arn:aws:iam::ACCOUNTNUMBER:role/ROLE-NAME',
    external_id = 'external_id',
    access_mode='RW'
)

# List all volumes available
volumes = api.volumes.query()

# List all files in volume
file_list = volume.list()

# The previous call only returns the first page of results, retrieving all
# files in a volume root directory is done by using 'all'. This does not
# include files in subdirectories
for volume_file in volume.list().all():
    print(volume_file)

# Subdirectories are stored in prefixes
prefixes = file_list.prefixes

# Files in first subdirectory
prefix = prefixes[0].prefix
file_list_sub = volume.list(prefix=prefix)
```

Import properties

When you import a file from a volume into a project on the Platform, you are importing a file from your cloud storage provider (Amazon Web Services or Google Cloud Storage) via the volume onto the Platform.

If successful, an alias will be created on the Platform. Aliases appear as files on the Platform and can be copied, executed, and modified as such. They refer back to the respective file on the given volume.

Each import has the following properties:

`href` - Import href on the API server.

`id` - Import identifier.

`source` - Source of the import, object of type `VolumeFile`, contains info on volume and file location on the volume.

`destination` - Destination of the import, object of type `ImportDestination`, containing info on project where the file was imported to and name of the file in the project.

`state` - State of the import. Can be *PENDING*, *RUNNING*, *COMPLETED* and *FAILED*.

`result` - If the import was completed, contains the result of the import - a `File` object.

`error` - Contains the `Error` object if the import failed.

`overwrite` - Whether the import was set to overwrite file at destination or not.

`started_on` - Contains the date and time when the import was started.

`finished_on` - Contains the date and time when the import was finished.

`preserve_folder_structure` - Whether to keep the exact source folder structure. The default value is true if the item being imported is a folder. Should not be used if you are importing a file.

`autorename` - Whether to automatically rename the item (by prefixing its name with an underscore and number) if another one with the same name already exists at the destination.

VolumeFile properties

`volume` - Volume ID from which to import the item.

`location` - Volume-specific location pointing to the file or folder to import. This location should be recognizable to the underlying cloud service as a valid key or path to the item. If the item being imported is a folder, its path should end with a `/`.

ImportDestination properties

`project` - The project in which to create the alias.

`parent` - The ID of the target folder to which the item should be imported. Should not be used together with project. If parent is used, the import will take place into the specified folder, within the project to which the folder belongs. If project is used, the items will be imported to the root of the project's files.

`name` - The name of the alias to create. This name should be unique to the project. If the name is already in use in the project, you should use the overwrite query parameter in this call to force any item with that name to be deleted before the alias is created. If name is omitted, the alias name will default to the last segment of the complete location (including the prefix) on the volume. Segments are considered to be separated with forward slashes `/`.

Import methods

Imports have the following methods:

- Refresh the import with data from the server: `reload()`
- Start an import by specifying the source and the destination of the import - `submit_import()`
- Delete the import - `delete()`

See the examples below for information on the arguments these methods take:

Examples

```
# Import a  file to a project
my_project = api.projects.get(id='my_project')
bucket_location = 'fastq/my_file.fastq'
imp = api.imports.submit_import(volume=volume_import, project=my_project,␣
↪location=bucket_location)
# Wait until the import finishes
while True:
      import_status = imp.reload().state
      if import_status in (ImportExportState.COMPLETED, ImportExportState.FAILED):
            break
      time.sleep(10)
# Continue with the import
if imp.state == ImportExportState.COMPLETED:
      imported_file = imp.result
```

Export properties

When you export a file from a project on the Platform into a volume, you are essentially writing to your cloud storage bucket on Amazon Web Services or Google Cloud Storage via the volume.

Note that the file selected for export must not be a public file or an alias. Aliases are objects stored in your cloud storage bucket which have been made available on the Platform.

The volume you are exporting to must be configured for read-write access. To do this, set the `access_mode` parameter to `RW` when creating or modifying a volume. Learn more about this from our Knowledge Center.

If an export command is successful, the original project file will become an alias to the newly exported object on the volume. The source file will be deleted from the Platform and, if no more copies of this file exist, it will no longer count towards your total storage price on the Platform. Once you export a file from the Platform to a volume, it is no longer part of the storage on the Platform and cannot be exported again.

Each export has the following properties:

`href` - Export href on the API server.

`id` - Export identifier.

`source` - Source of the export, object of type `File`

`destination` - Destination of the export, object of type `VolumeFile`, containing info on project where the file was imported to and name of the file in the project

`state` - State of the export. Can be *PENDING*, *RUNNING*, *COMPLETED* and *FAILED*.

`result` - If the export was completed, this contains the result of the import - a `File` object.

`error` - Contains the `Error` object if the export failed.

`overwrite` - Whether or not the export was set to overwrite the file at the destination.

`started_on` - Contains the date and time when the export was started.

`finished_on` - Contains the date and time when the export was finished.

Export methods

Exports have the following methods:

- Refresh the export with data from the server: `reload()`
- Submit export, by specifying source and destination of the import: `submit_import()`
- Delete the export: `delete()`

See the examples below for information on the arguments these methods take:

Examples

```
# Export a set of files to a volume
# Get files from a project
files_to_export = api.files.query(project=my_project).all()
# And export all the files to the output bucket
exports = []
for f in files_to_export:
      export = api.exports.submit_export(file=f, volume = volume_export, location=f.name)
```

(continues on next page)

(continued from previous page)

```
        exports.append(export)
# Wait for exports to finish:
num_exports = len(exports)
done = False

while not done:
        done_len = 0
        for e in exports:
                if e.reload().state in (ImportExportState.COMPLETED, ImportExportState.
↪FAILED):
                        done_len += 1
                time.sleep(10)
        if done_len == num_exports:
                done = True
```

1.3.16 Managing divisions

Enterprise access to the Platform is available on demand and allows companies or organizations to mimic their internal structure and hierarchy on the Seven Bridges Platform, thus facilitating simpler and more efficient collaboration.

Enterprise accounts are granted to users through their organizations. The organization associated with the Enterprise account may create user groups (Divisions and Teams) composed of Enterprise account holders, which are used to enable collaboration between members of the organization.

A Division on the Platform is a subgroup of users within an Enterprise. Use the Division entity to replicate the structure and hierarchy of an organization, such as departmental groups on the Platform. Divisions are created by Enterprise account administrators, but may be assigned Division administrator to manage the Division.

A Team on the Platform is a subgroup of a Division. Teams may be created by the Division administrator to simplify adding multiple members to a project simultaneously. One Division member can belong to multiple Teams.

The available methods for listing division are `query` and `get`, as for other objects:

```
# Query all divisions user belongs to (division authentication token required)
division_collection = api.divisions.query()
# Get a single division's information
division = api.divisions.get(id='division-id')
```

Division properties

`id` - Division ID.

`name` - Division name. Learn more about this in our Knowledge Center.

Division methods

Divisions have the following methods:

- Refresh the division with data from the server: `reload()`
- Get teams belonging to particular division `get_teams()`
- Get user members belonging to particular division `get_members()`

1.3.17 Managing apps

Managing apps (tools and workflows) with the sevenbridges-python library is simple. Apps on the Seven Bridges Platform and CGC are implemented using the Common Workflow Language (CWL) specification https://github.com/common-workflow-language/common-workflow-language.

Querying all apps or getting the details of a single app can be done in the same way as for other resources, using the `query()` and `get` methods. You can also invoke the following class-specific methods:

- `get_revision()` - Returns a specific app revision.
- `install_app()` - Installs your app on the server, using its CWL description.
- `create_revision()` - Creates a new revision of the specified app.
- `sync()` - Syncs the parent app changes with the current app instance.
- `copy()` - Copies the current app.

Note: Listing public apps can be achieved by invoking `api.apps.query(visibility='public')`

App properties

Each app has the following properties:

`href` - The URL of the app on the API server.

`id` - App identifier.

`name` - App name.

`project` - Identifier of the project that app is located in.

`revision` - App revision.

`raw` - Raw CWL description of the app.

App methods

- App only has class methods that were mentioned above.

1.3.18 Managing tasks

Tasks (pipeline executions) are easy to handle using the sevenbridges-python library. As with all resources you can `query()` your tasks, and `get()` a single task instance. You can also do much more. We will outline task properties and methods and show in the examples how easy is to run your first analysis using Python.

Task properties

`href` - Task URL on the API server.

`id` - Task identifier.

`name` - Task name.

`status` - Task status.

`project` - Identifier of the project that the task is located in.

`app` - The identifier of the app that was used for the task.

`type` - Task type.

`created_by` - Username of the task creator.

`executed_by`- Username of the task executor.

`batch` - Boolean flag: `True` for batch tasks, `False` for regular & child tasks.

`batch_by` - Batching criteria.

`batch_group` - Batch group assigned to the child task calculated from the `batch_by` criteria.

`batch_input` - Input identifier on to which to apply batching.

`parent` - Parent task for a batch child.

`end_time` - Task end time.

`execution_settings` - Execution settings for the task.

`output_location` - Location where task outputs will be stored.

`execution_status` - Task execution status.

`price` - Task cost.

`inputs` - Inputs that were submitted to the task.

`outputs` - Generated outputs from the task.

`origin` - Id of the entity that created the task, e.g. automation run, if task was created by an automation run

Note: Check the documentation on the Seven Bridges API and the CGC API for more details on batching criteria.

Task methods

The following class and instance methods are available for tasks:

- Create a task on the server and, optionally, run it: `create()`.
- Query tasks: `query()`.
- Get single task's information: `get()`.
- Abort a running task: `abort()`.
- Run a draft task: `run()`
- Delete a draft task from the server: `delete()`.
- Refresh the task object information with the date from the server: `refresh()`.
- Save task modifications to the sever: `save()`.
- Get task execution details: `get_execution_details()`.
- Get batch children if the task is a batch task: `get_batch_children()`.
- Clone task and optionally run it: `clone()`.

Task creation hints

- Both input files and parameters are passed the same way together in a single dictionary to `inputs`.
- **Supported execution settings are:**
  - *instance_type* - list or a string, can be 'AUTO' or an actual instance type, for example: ['c4.2xlarge;ebs-gp2;2000']
  - *max_parallel_instances* - Number of instances, >= 1

Querying tasks

- `api.tasks.query` always return an array of tasks. For single task inputs, use `api.tasks.query(project='my-project')[0]`.
- Queried tasks can be sorted with the `order_by` parameter. Supported fields are `created_time`, `start_time`, `name`, `end_time`, and `created_by`.
- Ordering can be specified with the `order` parameter. It is set to `desc` by default. Ascending order is set with `asc`.

Note: When querying running tasks it is recommended to use ordering, since the results are paginated and it is possible that some tasks will be duplicated or missed.

Task Examples

Single task

```
# Task name
name = 'my-first-task'

# Project in which I want to run a task.
project = 'my-username/my-project'

# App I want to use to run a task
app = 'my-username/my-project/my-app'

# Inputs
inputs = {}
inputs['FastQC-Reads'] = api.files.query(project='my-project', metadata={'sample': 'some-
↪sample'})

try:
    task = api.tasks.create(name=name, project=project, app=app, inputs=inputs, run=True)
except SbError:
    print('I was unable to run the task.')

# Task can also be ran by invoking .run() method on the draft task.
task.run()
```

Clone a task and run it with modification

Sometimes it is convenient to take an already executed task and change only some of the inputs, to then re-run it, using the clone method. Note: the clone method has the parameter *run=True* by default, so it is important to set it to *False* if modifications are needed.

```
# Getting the old task by id
old_task = api.tasks.get(old_task_id)

# clone it without launching it.
new_task = old_task.clone(run=False)

# Modify the inputs
new_task.inputs['somekey'] = "new value"

# Save the new task, and run it.
new_task.save()
new_task.run()
```

Batch task

```
# Task name
name = 'my-first-task'

# Project in which to run the task.
project = 'my-username/my-project'

# App to use to run the task
app = 'my-username/my-project/my-app'

# Inputs
inputs = {}
inputs['FastQC-Reads'] = api.files.query(project=project, metadata={'sample': 'some-
↪sample'})

# Specify that one task should be created per file (i.e. batch tasks by file).
batch_by = {'type': 'item'}


# Specify that the batch input is FastQC-Reads
batch_input = 'FastQC-Reads'

try:
    task = api.tasks.create(
        name=name, project=project, app=app, inputs=inputs,
        batch_input=batch_input, batch_by=batch_by, run=True
    )
except SbError:
    print('I was unable to run a batch task.')
```

Secondary files

Accessing task input/output secondary files is possible using the secondary_files property on the input/output itself.

Example for outputs:

```
task = api.tasks.get('<task_id>')
secondary_files = task.outputs['<output_name>'].secondary_files
```

1.3.19 Managing bulk operations

Bulk operations are supported for:

- Files
- Import jobs
- Export jobs

All bulk operations return a list of objects that contain a resource or an error. The state of any object can be checked with the `valid` property. If `valid` is set to True, `resource` is available, otherwise `error` is populated. Example:

```
response = api.files.bulk_get(files=files)
for record in response:
    if record.valid:
        print(record.resource)
    else:
        print(record.error)
```

The maximum number of resources that can be processed in a single bulk call is 100.

Files

The following operations are supported:

- `bulk_get()` - Retrieves multiple files.
- `bulk_edit()` - Modifies the existing information for specified files or add new information while preserving omitted parameters.
- `bulk_update()` - Sets new information for specified files, replacing all existing information and erasing omitted parameters.
- `bulk_delete()` - Deletes multiple files.

Retrieval and deletion are done by passing files (or file ids) in a list:

```
# Retrieve files
files = ['<FILE_ID>', '<FILE_ID>', '<FILE_ID>']
response = api.files.bulk_get(files=files)

# Delete files
files = [file1, file2, file3]
response = api.files.bulk_delete(files=files)
```

Editing and updating are done on file objects:

```
# Edit files
files = [edited_file1, edited_file2, edited_file3]
response = api.files.bulk_edit(files=files)

# Update files
files = [updated_file1, updated_file2, updated_file3]
response = api.files.bulk_update(files=files)
```

Properties that can be edited are `name`, `tags` and `metadata`.

Tasks

The following operations are supported:

- `bulk_get()` - Retrieves multiple tasks.

Retrieval is done by passing tasks (or task ids) in a list:

```
# Retrieve tasks
tasks = ['<TASK_ID>', '<TASK_ID>', '<TASK_ID>']
response = api.tasks.bulk_get(tasks=tasks)
```

Imports

The following operations are supported:

- `bulk_get()` - Retrieves multiple import jobs.
- `bulk_submit()` - Submits multiple import jobs.

Bulk retrieval, similarly to api.files.bulk_get(), requires a list of jobs:

```
# Retrieve imports
imports = ['<IMPORT_ID>', '<IMPORT_ID>', '<IMPORT_ID>']
response = api.imports.bulk_get(imports=imports)
```

Submitting in bulk can be done with a list of dictionaries with the required data for each job, for example:

```
volume = api.volumes.get('user/volume')
project = api.project.get('user/project')

# Submit import jobs
imports = [
    {
        'volume': volume,
        'location': '/data/example_file.txt',
        'project': project,
        'name': 'example_file.txt',
        'overwrite': False
    },
    {
        'volume': volume,
        'location': '/data/example_file_2.txt',
        'project': project,
        'name': 'example_file_2.txt',
        'overwrite': True
    }
]
response = api.imports.bulk_submit(imports=imports)
```

Exports

The following operations are supported:

- `bulk_get()` - Retrieves multiple export jobs.
- `bulk_submit()` - Submits multiple export jobs.

Bulk retrieval, similarly to api.files.bulk_get(), requires a list of jobs:

```
# Retrieve exports
exports = ['<EXPORT_ID>', '<EXPORT_ID>', '<EXPORT_ID>']
response = api.exports.bulk_get(exports=exports)
```

Submitting in bulk can be done with a list of dictionaries with the required data for each job, for example:

```
volume = api.volumes.get('user/volume')

# Find some files to export
files_to_export = list(api.files.query(project='user/my-project').all())

# Create exports list
# We will be exporting all the files in a project (root folder only) to
# the volume, with a location same as the name of the file

exports = []

for file in files_to_export:
      export = {
            'file':file,
            'volume': volume,
            'location':file.name
      }
      exports.append(export)

response = api.exports.bulk_submit(exports=exports, copy_only=False)
```

1.3.20 Managing automations

The following operations are supported for automations:

- `query()` - Query all automations
- `get()` - Get automation with the provided id
- `get_packages()` - Get all packages of an automation
- `get_package()` - Get package with provided id
- `add_package()` - Add a package to an automation
- `archive()` - Archive an automation
- `restore()` - Restore an archived automation
- `get_members()` - Get all members of an automation
- `get_member()` - Get details on a member of an automation
- `add_member()` - Add a member to an automation
- `remove_member()` - Remove member from an automation
- `get_runs()` - Get automation runs

The following operations are supported for automation packages:

- `query()` - Query all automation packages
- `get()` - Get automation package with the provided id
- `create()` - Create automation package
- `archive()` - Archive an package
- `restore()` - Restore an archived package

The following operations are supported for automation runs:

- `query()` - Query all automation runs
- `get()` - Get automation run with the provided id
- `create()` - Create and start new automation run
- `rerun()` - Reruns an existing automation run
- `stop()` - Stop an automation run
- `get_log()` - Get log file contents for an automation run
- `get_state()` - Get state file contents for an automation run

Properties

Each automation has the following properties:

`href` - The URL of the automation on the API server.

`id` - Automation identifier.

`name` - Automation name.

`description` - Automation description.

`owner` - Username of the user that owns the automation.

`created_by` - Username of the user that created the automation.

`created_on` - Date of the first automation creation.

`modified_by` - Username of the user that last modified the automation.

`modified_on` - Date of the last modification of the automation.

Each automation package has the following properties:

`id` - Automation package identifier.

`automation` - Identifier of the automation the package belongs to.

`version` - Automation package version.

`location` - Location identifier of uploaded automation package.

`created_by` - Username of the user that created the automation.

`created_on` - Date of the automation package creation.

`archived` - Flag indicating wheather automation package is arhived or not.

`custom_url` - Link to custom frontend page.

`python` - Python version to run package with.

Each automation run has the following properties:

`href` - The URL of the automation run on the API server.

`id` - Automation identifier.

`automation` - Automation identifier of the automation the run belongs to.

`package` - Automation package identifier of the package the run belongs to.

`inputs` - Automation run input dictionary.

`settings` - Automation run settings override dictionary.

`created_on` - Date of the first automation run creation.

`start_time` - Date of the automation run start.

`end_time` - Date of the automation run end.

`resumed_from` - Automation run identifier of the automation run the run resumed from.

`created_by` - Username of the user that created the automation run.

`status` - Current status of the automation run.

`message` - Message output of the automation run.

`execution_details` - Execution details of the automation run.

Examples

```
# Query automation runs with name parameter
api.automation_runs.query(name='automation_run_name')

# List all automations
automations = api.automations.query()

# Get details of an automation
automation = api.automations.get('automation_id')

# Create automation package from uploaded code package
automation_package = api.automation_packages.create('automation_id', 'version',
↪'location_id'):

# Add a code package to automation from local file
automation.add_package('version', 'file_path', schema={})

# Get automation package details
automation_package = api.automation_packages.get('package_id')

# Get automation runs with name parameter for existing automation
automation.get_runs(name='automation_run_name')

# List all packages that belong to an automation
packages = automation.get_packages()

# List all members that belong to an automation
members = automation.get_members()

# Get details of an automation member
member = automation.get_member('member_username')

# Add new member to automation
username = 'new_member_username'
permissions = {
    'read': True,
    'write': True,
```

(continues on next page)

(continued from previous page)

```
    'copy': True,
    'execute': True,
    'admin': True,
}
new_member = automation.add_member(username, permissions)

# Edit member permissions
new_member.permissions['admin'] = False
new_member.save()

# Remove member from automation
automation.remove_member('automation_member')

# List automation runs
runs = api.automation_runs.query()

# Get details of an automation run
run = api.automation_runs.get('automation_run_id')

# Start a new automation run
new_run = api.automation_runs.create(
    package='package_id',
    name='automation_run_name',
    inputs={
        'x': 1,
        'y': 2,
        'z': 3
    }
)

# Stop an automation run
new_run.stop()

# Get automation run log
state = run.log()

# Get automation run state
state = run.state()
```

1.3.21 Managing async operations

The following operations are supported for async operations runs:

- `list_file_jobs()` - Query all async jobs for files.
- `get_copy_files_job()` - Get the details of an asynchronous bulk file copy job.
- `get_delete_files_job()` - Get the details of an asynchronous bulk file delete job.
- `get_file_move_job()` - Get the details of an asynchronous bulk file move job.
- `get_result()` - Parse results of a job as a bulk response.

- `file_bulk_copy()` - Perform a bulk copy operation of files and folders. Any underlying folder structure will be preserved.
- `file_bulk_move()` - Perform a bulk move operation of files and folders. Any underlying folder structure will be preserved.
- `file_bulk_delete()` - Perform a bulk delete operation of files and folders. Deleting folders which aren't empty is allowed.

Properties

Each async job has the following properties:

`id` - Async job identifier.

`type` - The type of job, which is COPY in the case of copying files, MOVE in case of moving files and DELETE in case of deleting files.

`state` - Current job state (RUNNING, FINISHED, SUBMITTED, RESOLVING)

`result` - The result of the copy job.

`total_files` - The total number of files that were processed during the job.

`failed_files` - The number of files that failed to copy.

`completed_files` - The number of files that were successfully copied.

`started_on` - The time and date the copy job started.

`finished_on` - The time and date the copy job has finished.

Examples

```
# Query all file jobs
all_jobs = api.async_jobs.list_file_jobs().all()

# Get file copy job by id
copy_job = api.async_jobs.get_file_copy_job(id='copy_job_id')

# Get file delete job by id
delete_job = api.async_jobs.get_file_delete_job(id='delete_job_id')

# Parse job results to bulk format
copy_job_results = copy_job.get_result()

# Check validity of each result in bulk format
for result in copy_job_results:
    if result.valid:
        print(result.resource)
    else:
        print(result.error)

# Start bulk file copy job
files = [
    {
        'file': 'file_id_1',
```

(continues on next page)

(continued from previous page)

```
        'parent': 'parent_id',
        'name': 'new_name_1',
    },
    {
        'file': 'file_id_2',
        'parent': 'parent_id',
        'name': 'new_name_2',
    },
]
new_copy_job = api.async_jobs.file_bulk_copy(files=files)

# Start bulk file delete job
files = [
    {
        'file': 'file_id_1'
    },
    {
        'file': 'file_id_2'
    },
]
new_delete_job = api.async_jobs.file_bulk_delete(files=files)

# Start bulk file move job
    files = [
        {
            'file': 'file_id_1',
            'project': project_id,
            'name': 'name_1',
        },
        {
            'file': 'file_id_2',
            'project': project_id,
            'name': 'name_2',
        },
        {
            'file': 'file_id_3',
            'project': project_id,
            'name': 'name_2',
        }
    ]
    new_move_job = api.async_jobs.file_bulk_move(files=files)
```

1.3.22 Managing DRS bulk imports

The following operations are supported for drs bulk imports:

- `bulk_get()` - Retrieve DRS bulk import details.
- `bulk_submit()` - Submit DRS bulk import.

Properties

Each drs bulk import has the following properties:

`href` - URI of the import job

`id` - ID of the import job.

`result` - `File` or `Error` object for each of the items in the bulk import job.

`state` - State of the import job. Can be *PENDING*, *RUNNING*, *FINISHED* and *SUBMITTED RESOLVING*.

`started_on` - Contains the date and time when the import job started.

`finished_on` - Contains the date and time when the import job finished.

`result_files` - List of files that were successfully imported.

Examples

```
project = api.project.get('user/project')
import_data = [
    {
       "name": "filename.fasta",
       "drs_uri": "drs://caninedc.org/01349ad3-6008-426f-a17c-dasdf131e",
       "parent": "568cf5dce4asd232bc0462060",
       "metadata": {"study_id": "123", "cohort": 2}
    },
    {
       "name": "filename_2.fasta",
       "drs_uri": "drs://caninedc.org/01349ad3-6008-426f-a17c-asf2323saf",
       "project": project,
       "metadata": {"study_id": "123", "cohort": 3}
    }
]

# Submit import jobs and wait for it to be completed
import_job = api.imports.bulk_submit(
    imports=import_data,
    tags=['tag1', 'tag2'],
    conflict_resolution='OVERWRITE'
)
while import_job.state != 'FINISHED':
    time.sleep(30)
    import_job.reload()

imported_files = import_job.result_files
```

(continues on next page)

(continued from previous page)

```
# Submit import jobs and check later for its state
import_job = api.drs_imports.bulk_submit(imports=import_data)
import_id = import_job.id
import_job = api.drs_imports.bulk_get(import_job_id=import_id)
if import_job.state == 'FINISHED':
    print("Imports finished!")
```

1.4 sevenbridges package

1.4.1 sevenbridges-python

copyright
2020 Seven Bridges Genomics Inc.

license
Apache 2.0

class sevenbridges.**Api**(*url=None*, *token=None*, *oauth_token=None*, *config=None*, *timeout=None*, *download_max_workers=16*, *upload_max_workers=16*, *proxies=None*, *error_handlers=None*, *advance_access=False*, *pool_connections=10*, *pool_maxsize=100*, *pool_block=True*, *max_parallel_requests=100*, *retry_count=6*, *backoff_factor=1*, *debug=False*)

Bases: *HttpClient*

Api aggregates all resource classes into single place

actions
alias of *Actions*

apps
alias of *App*

async_jobs
alias of *AsyncJob*

automation_packages
alias of *AutomationPackage*

automation_runs
alias of *AutomationRun*

automations
alias of *Automation*

billing_groups
alias of *BillingGroup*

datasets
alias of *Dataset*

divisions
alias of *Division*

drs_imports
alias of *DRSImportBulk*

endpoints
alias of *Endpoints*

exports
alias of *Export*

files
alias of *File*

imports
alias of *Import*

invoices
alias of *Invoice*

markers
alias of *Marker*

projects
alias of *Project*

rate_limit
alias of *RateLimit*

tasks
alias of *Task*

teams
alias of *Team*

users
alias of *User*

volumes
alias of *Volume*

class sevenbridges.**App**(***kwargs*)

Bases: *Resource*

Central resource for managing apps.

copy(*project*, *name=None*, *strategy=None*, *use_revision=False*, *api=None*)

Copies the current app. :param project: Destination project. :param name: Destination app name. :param strategy: App copy strategy. :param use_revision: Copy from set app revision. :param api: Api instance. :return: Copied App object.

Copy strategies

clone copy all revisions and continue getting updates form the original app (default method when the key is omitted)

direct copy only the latest revision and get the updates from this point on

clone_direct copy the app like the direct strategy, but keep all revisions

transient copy only the latest revision and continue getting
updates from the original app

classmethod create_revision(*id*, *revision*, *raw*, *api=None*)

Create a new app revision. :param id: App identifier. :param revision: App revision. :param raw: Raw cwl object. :param api: Api instance. :return: App object.

deepcopy()

equals(*other*)

classmethod get_revision(*id*, *revision*, *api=None*)

Get app revision. :param id: App identifier. :param revision: App revision :param api: Api instance. :return: App object.

href = None

property id

classmethod install_app(*id*, *raw*, *api=None*, *raw_format=None*)

Installs and app. :param id: App identifier. :param raw: Raw cwl data. :param api: Api instance. :param raw_format: Format of raw app data being sent, json by default :return: App object.

name = None

project = None

classmethod query(*project=None*, *visibility=None*, *q=None*, *id=None*, *offset=None*, *limit=None*, *api=None*)

Query (List) apps. :param project: Source project. :param visibility: private|public for private or public apps. :param q: List containing search terms. :param id: List contains app ids. Fetch apps with specific ids. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: collection object

raw = None

revision = None

sync()

Syncs the parent app changes with the current app instance. :return: Synced App object.

class sevenbridges.**AppCopyStrategy**

Bases: object

CLONE = 'clone'

CLONE_DIRECT = 'clone_direct'

DIRECT = 'direct'

TRANSIENT = 'transient'

class sevenbridges.**AppRawFormat**

Bases: object

JSON = 'json'

YAML = 'yaml'

class sevenbridges.**AsyncFileOperations**

Bases: object

COPY = 'copy'

DELETE = 'delete'

MOVE = 'move'

class sevenbridges.**AsyncJob**(***kwargs*)

Bases: *Resource*

Central resource for managing async jobs

completed_files = None

deepcopy()

equals(*other*)

failed_files = None

classmethod file_bulk_copy(*files*, *api=None*)

classmethod file_bulk_delete(*files*, *api=None*)

classmethod file_bulk_move(*files*, *api=None*)

finished_on = None

classmethod get_file_copy_job(*id*, *api=None*)

Retrieve file copy async job :param id: Async job identifier :param api: Api instance :return:

classmethod get_file_delete_job(*id*, *api=None*)

Parameters

- **id** – Async job identifier
- **api** – Api instance

Returns

classmethod get_file_move_job(*id*, *api=None*)

Retrieve file move async job :param id: Async job identifier :param api: Api instance :return:

get_result(*api=None*)

Get async job result in bulk format :return: List of AsyncFileBulkRecord objects

id = None

classmethod list_file_jobs(*offset=None*, *limit=None*, *api=None*)

Query (List) async jobs :param offset: Pagination offset :param limit: Pagination limit :param api: Api instance :return: Collection object

result = None

started_on = None

state = None

total_files = None

type = None

class sevenbridges.**AsyncJobStates**

Bases: object

FINISHED = 'FINISHED'

RESOLVING = 'RESOLVING'

RUNNING = 'RUNNING'

SUBMITTED = 'SUBMITTED'

class sevenbridges.**Automation**(***kwargs*)

Bases: *Resource*

Central resource for managing automations.

add_member(*user*, *permissions*, *api=None*)

Add member to the automation :param user: Member username :param permissions: Member permissions :param api: sevenbridges Api instance :return: AutomationMember object

add_package(*version*, *file_path*, *schema*, *file_name=None*, *python=None*, *retry_count=6*, *timeout=300*, *part_size=None*, *api=None*)

Add a code package to automation template. :param version: The code package version. :param file_path: Path to the code package file to be uploaded. :param schema: IO schema for main step of execution. :param part_size: Size of upload part in bytes. :param file_name: Optional file name. :param python: Version of Python to execute Code Package with. Allowed values are '3.6', '3.7', '3.8'. Default is '3.6'. :param retry_count: Upload retry count. :param timeout: Timeout for s3/google session. :param api: sevenbridges Api instance. :return: AutomationPackage

add_team_member(*team*, *permissions*, *api=None*)

Add team member to the automation :param team: Team object or team id :param permissions: Member permissions :param api: sevenbridges Api instance :return: AutomationMember object

archive(**args*, ***kwargs*)

archived = None

billing_group = None

classmethod create(*name*, *description=None*, *billing_group=None*, *secret_settings=None*, *project_based=None*, *memory_limit=None*, *api=None*)

Create a automation template. :param name: Automation name. :param billing_group: Automation billing group. :param description: Automation description. :param secret_settings: Automation settings. :param project_based: Create project based automation template. :param memory_limit: Memory limit in MB. :param api: Api instance. :return:

created_by = None

created_on = None

deepcopy()

description = None

equals(*other*)

get_member(*username*, *api=None*)

Return specified automation member :param username: Member username :param api: sevenbridges Api instance. :return: AutomationMember object

get_members(*offset=None*, *limit=None*, *api=None*)

Return list of automation members :param offset: Pagination offset. :param limit: Pagination limit. :param api: sevenbridges Api instance. :return: AutomationMember collection

classmethod get_package(*package*, *api=None*)

Return specified automation member :param package: Automation Package Id :param api: sevenbridges Api instance. :return: AutomationMember object

get_packages(*offset=None*, *limit=None*, *api=None*)

Return list of packages that belong to this automation :param offset: Pagination offset. :param limit: Pagination limit. :param api: sevenbridges Api instance. :return: AutomationPackage collection

get_runs(*package=None*, *status=None*, *name=None*, *created_by=None*, *created_from=None*, *created_to=None*, *project_id=None*, *order_by=None*, *order=None*, *offset=None*, *limit=None*, *api=None*)

Query automation runs that belong to this automation :param package: Package id :param status: Run status :param name: Automation run name :param created_by: Username of member that created the run :param created_from: Date the run was created after :param created_to: Date the run was created before :param project_id: Search runs by project id, if run is project based :param order_by: Property by which to order results :param order: Ascending or Descending ("asc" or "desc") :param offset: Pagination offset. :param limit: Pagination limit. :param api: sevenbridges Api instance :return: AutomationRun collection

href = None

id = None

memory_limit = None

modified_by = None

modified_on = None

name = None

owner = None

project_based = None

classmethod query(*name=None*, *include_archived=False*, *project_based=None*, *offset=None*, *limit=None*, *api=None*)

Query (List) automations. :param name: Automation name. :param include_archived: Include archived automations also :param project_based: Search project based automations :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: collection object

remove_member(*user*, *api=None*)

Remove a member from the automation :param user: Member username :param api: sevenbridges Api instance :return: None

remove_team_member(*team*, *api=None*)

Remove a team member from the automation :param team: Team object or team id :param api: sevenbridges Api instance :return: None

restore(**args*, ***kwargs*)

save(**args*, ***kwargs*)

secret_settings = None

class sevenbridges.**AutomationMember**(***kwargs*)

Bases: *Resource*

Central resource for managing automation members.

classmethod add(*user*, *permissions*, *automation*, *api=None*)

Add a member to the automation. :param user: Member username :param permissions: Permissions dictionary. :param automation: Automation object or id :param api: sevenbridges Api instance :return: Automation member object.

classmethod add_team(*team*, *permissions*, *automation*, *api=None*)

Add a team as a member to the automation. :param team: Team object or team id :param permissions: Permissions dictionary. :param automation: Automation object or id :param api: sevenbridges Api instance :return: Automation member object.

deepcopy()

email = None

equals(*other*)

classmethod get(*id*, *automation*, *api=None*)

Fetches the resource from the server. :param id: Automation member username :param automation: Automation id or object :param api: sevenbridges Api instance. :return: AutomationMember object.

href = None

id = None

name = None

permissions

classmethod query(*automation=None*, *offset=None*, *limit=None*, *api=None*)

Query (List) apps. :param automation: Automation id. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: collection object

classmethod remove(*user*, *automation*, *api=None*)

Remove a member from the automation. :param user: Member username :param automation: Automation id :param api: sevenbridges Api instance :return: None

classmethod remove_team(*team*, *automation*, *api=None*)

Remove a team member from the automation. :param team: Team object or team id :param automation: Automation id :param api: sevenbridges Api instance :return: None

save(**args*, ***kwargs*)

type = None

username = None

class sevenbridges.**AutomationPackage**(***kwargs*)

Bases: *Resource*

Central resource for managing automation packages.

archive(**args*, ***kwargs*)

archived = None

automation = None

classmethod create(*automation*, *version*, *location*, *schema*, *memory_limit=None*, *python=None*, *api=None*)

Create a code package. :param automation: Automation id. :param version: File ID of the uploaded code package. :param location: The code package version. :param schema: IO schema for main step of execution. :param python: Version of Python to execute Code Package with. Allowed values are '3.6', '3.7', '3.8'. Default is '3.6'. :param memory_limit: Memory limit in MB. :param api: Api instance. :return:

created_by = None

created_on = None

custom_url = None

deepcopy()

equals(*other*)

id = None

location = None

memory_limit = None

python = None

classmethod query(*automation*, *offset=None*, *limit=None*, *api=None*)

Query (List) automation packages. :param automation: Automation id. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: collection object

restore(**args*, ***kwargs*)

save(**args*, ***kwargs*)

schema = None

version = None

class sevenbridges.**AutomationRun**(***kwargs*)

Bases: *Resource*

Central resource for managing automation runs.

automation

classmethod create(*package*, *inputs=None*, *settings=None*, *resume_from=None*, *name=None*, *secret_settings=None*, *memory_limit=None*, *api=None*)

Create and start a new run. :param package: Automation package id :param inputs: Input dictionary :param settings: Settings override dictionary :param resume_from: Run to resume from :param name: Automation run name :param secret_settings: dict to override secret_settings from automation template :param memory_limit: Memory limit in MB. :param api: sevenbridges Api instance :return: AutomationRun object

created_by = None

created_on = None

deepcopy()

end_time = None

equals(*other*)

execution_details = None

get_log_file(*api=None*)

Retrieve automation run log. :param api: sevenbridges Api instance :return: Log string

get_state(*api=None*)

Retrieve automation run state. :param api: sevenbridges Api instance :return: State file json contents as string

href = None

id = None

inputs = None

memory_limit = None

message = None

name = None

outputs = None

package

project_id = None

classmethod query(*automation=None*, *package=None*, *status=None*, *name=None*, *created_by=None*, *created_from=None*, *created_to=None*, *project_id=None*, *order_by=None*, *order=None*, *offset=None*, *limit=None*, *api=None*)

Query (List) automation runs. :param name: Automation run name :param automation: Automation template :param package: Package :param status: Run status :param created_by: Username of user that created the run :param order_by: Property by which to order results :param order: Ascending or descending ("asc" or "desc") :param created_from: Date the run is created after :param created_to: Date the run is created before :param project_id: Id of project if Automation run is project based :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: collection object

classmethod rerun(*id*, *package=None*, *inputs=None*, *settings=None*, *resume_from=None*, *name=None*, *secret_settings=None*, *merge=True*, *api=None*)

Create and start rerun of existing automation. :param id: Automation id to rerun :param package: Automation package id :param inputs: Input dictionary :param settings: Settings override dictionary :param resume_from: Run to resume from :param name: Automation run name :param secret_settings: dict to override secret_settings from automation template :param merge: merge settings and inputs of run :param api: sevenbridges Api instance :return: AutomationRun object

resumed_from = None

save(**args*, ***kwargs*)

settings = None

start_time = None

status = None

stop(*api=None*)

Stop automation run. :param api: sevenbridges Api instance. :return: AutomationRun object

class sevenbridges.**AutomationRunActions**

Bases: object

RERUN = 'rerun'

STOP = 'stop'

class sevenbridges.**AutomationStatus**

Bases: object

ABORTED = 'ABORTED'

CREATED = 'CREATED'

FAILED = 'FAILED'

FINISHED = 'FINISHED'

QUEUED_FOR_EXECUTION = 'QUEUED_FOR_EXECUTION'

QUEUED_FOR_TERMINATION = 'QUEUED_FOR_TERMINATION'

RUNNING = 'RUNNING'

SENT_TO_EXECUTION = 'SENT_TO_EXECUTION'

terminal_states = ['FINISHED', 'FAILED', 'ABORTED']

exception sevenbridges.**BadRequest**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

class sevenbridges.**BillingGroup**(***kwargs*)

Bases: *Resource*

Central resource for managing billing groups.

analysis_breakdown(*date_from=None*, *date_to=None*, *invoice_id=None*, *fields=None*, *offset=None*, *limit=None*)

Get Billing group analysis breakdown for the current billing group.

balance

deepcopy()

disabled = None

egress_breakdown(*date_from=None*, *date_to=None*, *invoice_id=None*, *fields=None*, *offset=None*, *limit=None*)

Get Billing group egress breakdown for the current billing group.

equals(*other*)

href = None

id = None

name = None

owner = None

pending = None

classmethod query(*offset=None*, *limit=None*, *api=None*)

Query (List) billing group. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object. :param api: Api instance.

storage_breakdown(*date_from=None*, *date_to=None*, *invoice_id=None*, *fields=None*, *offset=None*, *limit=None*)

Get Billing group storage breakdown for the current billing group.

type = None

class sevenbridges.**BulkRecord**(***kwargs*)

Bases: *Resource*

deepcopy()

equals(*other*)

error

classmethod parse_records(*response*, *api=None*)

resource

property valid

class sevenbridges.**Config**(*profile=None*, *proxies=None*, *advance_access=None*)

Bases: object

Utility configuration class.

exception sevenbridges.**Conflict**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

class sevenbridges.**DRSImportBulk**(***kwargs*)

Bases: *Resource*

Central resource for managing DRS imports.

classmethod bulk_get(*import_job_id*, *api=None*)

Retrieve DRS bulk import details :param import_job_id: Import id to be retrieved. :param api: Api instance. :return: DRSImportBulk object.

classmethod bulk_submit(*imports*, *tags=None*, *conflict_resolution='SKIP'*, *api=None*)

Submit DRS bulk import :param imports: List of dicts describing a wanted import. :param tags: list of tags to be applied. :param conflict_resolution: Type of file naming conflict resolution. :param api: Api instance. :return: DRSImportBulk object.

deepcopy()

equals(*other*)

finished_on = None

href = None

id = None

result

property result_files

Retrieve files that were successfully imported. :return: List of File objects

started_on = None

state = None

class sevenbridges.**Dataset**(***kwargs*)

Bases: *Resource*

Central resource for managing datasets.

add_member(*username*, *permissions*, *api=None*)

Add member to a dataset :param username: Member username :param permissions: Permissions dict :param api: Api instance :return: New member instance

deepcopy()

description = None

equals(*other*)

get_member(*username*, *api=None*)

Retrieve dataset member :param username: Member name :param api: Api instance :return: Member object

get_members(*api=None*)

Retrieve dataset members :param api: Api instance :return: Collection object

classmethod get_owned_by(*username*, *api=None*)

Query (List) datasets by owner :param api: Api instance :param username: Owner username :return: Collection object

href = None

id = None

name = None

classmethod query(*visibility=None*, *api=None*)

Query (List) datasets :param visibility: If provided as 'public', retrieves public datasets :param api: Api instance :return: Collection object

remove_member(*member*, *api=None*)

Remove member from a dataset :param member: Member username :param api: Api instance :return: None

save(**args*, ***kwargs*)

class sevenbridges.**Division**(***kwargs*)

Bases: *Resource*

Central resource for managing divisions.

deepcopy()

equals(*other*)

get_members(*role=None*, *offset=None*, *limit=None*)

get_teams(*offset=None*, *limit=None*)

href = None

id = None

name = None

classmethod query(*offset=None*, *limit=None*, *api=None*)

Query (List) divisions.

Parameters

- **offset** – Pagination offset.
- **limit** – Pagination limit.
- **api** – Api instance.

Returns

Collection object.

class sevenbridges.**DivisionRole**

Bases: object

ADMIN = 'admin'

EXTERNAL_COLLABORATOR = 'external_collaborator'

MEMBER = 'member'

class sevenbridges.**Endpoints**(***kwargs*)

Bases: *Resource*

Central resource for managing Endpoints.

action_url = None

apps_url = None

billing_url = None

deepcopy()

equals(*other*)

files_url = None

classmethod get(*api=None*, ***kwargs*)

Get api links. :param api: Api instance. :return: Endpoints object.

projects_url = None

rate_limit_url = None

tasks_url = None

upload_url = None

user_url = None

users_url = None

exception sevenbridges.**ExecutionDetailsInvalidTaskType**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

class sevenbridges.**Export**(***kwargs*)

Bases: *Resource*

Central resource for managing exports.

classmethod bulk_get(*exports*, *api=None*)

Retrieve exports in bulk. :param exports: Exports to be retrieved. :param api: Api instance. :return: list of ExportBulkRecord objects.

classmethod bulk_submit(*exports*, *copy_only=False*, *api=None*)

Create exports in bulk. :param exports: List of dicts describing a wanted export. :param copy_only: If true files are kept on SevenBridges bucket. :param api: Api instance. :return: list of ExportBulkRecord objects.

deepcopy()

destination

equals(*other*)

error

finished_on = None

href = None

id = None

overwrite = None

properties

classmethod query(*volume=None*, *state=None*, *offset=None*, *limit=None*, *api=None*)

Query (List) exports. :param volume: Optional volume identifier. :param state: Optional import sate. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

property result

property source

started_on = None

state = None

classmethod submit_export(*file*, *volume*, *location*, *properties=None*, *overwrite=False*, *copy_only=False*, *api=None*)

Submit new export job. :param file: File to be exported. :param volume: Volume identifier. :param location: Volume location. :param properties: Properties dictionary. :param overwrite: If true it will overwrite file if exists :param copy_only: If true files are kept on SevenBridges bucket. :param api: Api Instance. :return: Export object.

class sevenbridges.**File**(***kwargs*)

Bases: *Resource*

Central resource for managing files.

FOLDER_TYPE = 'folder'

classmethod bulk_delete(*files*, *api=None*)

Delete files with specified ids in bulk :param files: Files to be deleted. :param api: Api instance. :return: List of FileBulkRecord objects.

classmethod bulk_edit(*files*, *api=None*)

This call edits the details for multiple specified files. Use this call to modify the existing information for the files or add new information while preserving omitted parameters. For each of the specified files, the call edits its name, tags and metadata. :param files: List of file instances. :param api: Api instance. :return: List of FileBulkRecord objects.

classmethod bulk_get(*files*, *api=None*)

Retrieve files with specified ids in bulk :param files: Files to be retrieved. :param api: Api instance. :return: List of FileBulkRecord objects.

classmethod bulk_update(*files*, *api=None*)

This call updates the details for multiple specified files. Use this call to set new information for the files, thus replacing all existing information and erasing omitted parameters. For each of the specified files, the call sets a new name, new tags and metadata. :param files: List of file instances. :param api: Api instance. :return: List of FileBulkRecord objects.

content(*path=None*, *overwrite=True*, *encoding='utf-8'*)

Downloads file to the specified path or as temporary file and reads the file content in memory. Should not be used on very large files. :param path: Path for file download If omitted tmp file will be used. :param overwrite: Overwrite file if exists locally :param encoding: File encoding, by default it is UTF-8 :return: File content.

copy(*project*, *name=None*)

Copies the current file. :param project: Destination project. :param name: Destination file name. :return: Copied File object.

copy_to_folder(*parent*, *name=None*, *api=None*)

Copy file to folder :param parent: Folder to copy file to :param name: New file name :param api: Api instance :return: New file instance

classmethod create_folder(*name*, *parent=None*, *project=None*, *api=None*)

Create a new folder :param name: Folder name :param parent: Parent folder :param project: Project to create folder in :param api: Api instance :return: New folder

created_on = None

deepcopy()

download(*path*, *retry=6*, *timeout=300*, *chunk_size=None*, *wait=True*, *overwrite=False*)

Downloads the file and returns a download handle. Download will not start until .start() method is invoked. :param path: Full path to the new file. :param retry: Number of retries if error occurs during download. :param timeout: Timeout for http requests. :param chunk_size: Chunk size in bytes. :param wait: If true will wait for download to complete. :param overwrite: If True will silently overwrite existing file. :return: Download handle.

download_info()

Fetches download information containing file url that can be used to download file. :return: Download info object.

equals(*other*)

href = None

id = None

is_folder()

list_files(*offset=None*, *limit=None*, *api=None*, *cont_token=None*)

List files in a folder :param api: Api instance :param offset: Pagination offset :param limit: Pagination limit :param cont_token: Pagination continuation token :return: List of files

metadata

modified_on = None

move_to_folder(*parent*, *name=None*, *api=None*)

Move file to folder :param parent: Folder to move file to :param name: New file name :param api: Api instance :return: New file instance

name = None

origin

parent = None

project = None

classmethod query(*project=None*, *names=None*, *metadata=None*, *origin=None*, *tags=None*, *offset=None*, *limit=None*, *dataset=None*, *api=None*, *parent=None*, *cont_token=None*)

Query (List) files, requires project or dataset :param project: Project id :param names: Name list :param metadata: Metadata query dict :param origin: Origin query dict :param tags: List of tags to filter on :param offset: Pagination offset :param limit: Pagination limit :param dataset: Dataset id :param api: Api instance. :param parent: Folder id or File object with type folder :param cont_token: Pagination continuation token :return: Collection object.

reload()

Refreshes the file with the data from the server.

save(**args*, ***kwargs*)

classmethod search(*query*, *cont_token=None*, *limit=None*, *api=None*)

Search files by a query. :param query: Query written in SBG query language. :param cont_token: Continuation token value. :param limit: Limit value. :param api: Api instance.

property secondary_files

size = None

storage

stream(*part_size=32768*)

Creates an iterator which can be used to stream the file content. :param part_size: Size of the part in bytes. Default 32KB :return Iterator

tags = None

type = None

classmethod upload(*path*, *project=None*, *parent=None*, *file_name=None*, *overwrite=False*, *retry=6*, *timeout=300*, *part_size=None*, *wait=True*, *api=None*)

Uploads a file using multipart upload and returns an upload handle if the wait parameter is set to False. If wait is set to True it will block until the upload is completed.

Parameters

- **path** – File path on local disc.
- **project** – Project identifier
- **parent** – Parent folder identifier
- **file_name** – Optional file name.
- **overwrite** – If true will overwrite the file on the server.
- **retry** – Number of retries if error occurs during upload.
- **timeout** – Timeout for http requests.
- **part_size** – Part size in bytes.
- **wait** – If true will wait for upload to complete.
- **api** – Api instance.

class sevenbridges.**FileStorageType**

Bases: object

PLATFORM = 'PLATFORM'

VOLUME = 'VOLUME'

exception sevenbridges.**Forbidden**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

class sevenbridges.**Import**(***kwargs*)

Bases: *Resource*

Central resource for managing imports.

autorename = None

classmethod bulk_get(*imports*, *api=None*)

Retrieve imports in bulk :param imports: Imports to be retrieved. :param api: Api instance. :return: List of ImportBulkRecord objects.

classmethod bulk_submit(*imports*, *api=None*)

Submit imports in bulk :param imports: List of dicts describing a wanted import. :param api: Api instance. :return: List of ImportBulkRecord objects.

deepcopy()

destination

equals(*other*)

error

finished_on = None

href = None

id = None

overwrite = None

preserve_folder_structure = None

classmethod query(*project=None*, *volume=None*, *state=None*, *offset=None*, *limit=None*, *api=None*)

Query (List) imports. :param project: Optional project identifier. :param volume: Optional volume identifier. :param state: Optional import sate. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

property result

source

started_on = None

state = None

classmethod submit_import(*volume*, *location*, *project=None*, *name=None*, *overwrite=False*, *properties=None*, *parent=None*, *preserve_folder_structure=True*, *autorename=False*, *api=None*)

Submits new import job. :param volume: Volume identifier. :param location: Volume location. :param project: Project identifier. :param name: Optional file name. :param overwrite: If true it will overwrite file if exists. :param properties: Properties dictionary. :param parent: The ID of the target folder to which the item should be imported. Should not be used together with project. :param preserve_folder_structure: Whether to keep the exact source folder structure. The default value is true if the item being imported is a folder. Should not be used if you are importing a file. :param autorename: Whether to automatically rename the item (by prefixing its name with an underscore and number) if another one with the same name already exists at the destination. :param api: Api instance. :return: Import object.

class sevenbridges.**ImportExportState**

Bases: object

COMPLETED = 'COMPLETED'

FAILED = 'FAILED'

PENDING = 'PENDING'

RUNNING = 'RUNNING'

class sevenbridges.**Invoice**(***kwargs*)

Bases: *Resource*

Central resource for managing invoices.

analysis_costs

deepcopy()

equals(*other*)

href = None

id = None

invoice_period

pending = None

classmethod query(*offset=None*, *limit=None*, *api=None*)

Query (List) invoices. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

storage_costs

total

exception sevenbridges.**LocalFileAlreadyExists**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

class sevenbridges.**Marker**(***kwargs*)

Bases: *Resource*

chromosome = None

classmethod create(*file*, *name*, *position*, *chromosome*, *private=True*, *api=None*)

Create a marker on a file. :param file: File object or identifier. :param name: Marker name. :param position: Marker position object. :param chromosome: Chromosome number. :param private: Whether the marker is private or public. :param api: Api instance. :return: Marker object.

created_by = None

created_time = None

deepcopy()

equals(*other*)

file = None

href = None

id = None

name = None

position

classmethod query(*file*, *offset=None*, *limit=None*, *api=None*)

Queries genome markers on a file. :param file: Genome file - Usually bam file. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

save(**args*, ***kwargs*)

class sevenbridges.**Member**(***kwargs*)

Bases: *Resource*

Central resource for managing members. This resource is reused on both projects and volumes.

deepcopy()

email = None

equals(*other*)

href = None

id = None

permissions

save(**args*, ***kwargs*)

type = None

username = None

exception sevenbridges.**MethodNotAllowed**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.**NotFound**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.**PaginationError**(*message*)

Bases: *SbgError*

class sevenbridges.**PartSize**

Bases: object

DOWNLOAD_MINIMUM_PART_SIZE = 5242880

GB = 1073741824

KB = 1024

MAXIMUM_OBJECT_SIZE = 5497558138880

MAXIMUM_TOTAL_PARTS = 10000

MAXIMUM_UPLOAD_SIZE = 5368709120

MB = 1048576

TB = 1099511627776

UPLOAD_MINIMUM_PART_SIZE = 5242880

UPLOAD_RECOMMENDED_SIZE = 33554432

class sevenbridges.**Permissions**(***kwargs*)

Bases: *CompoundMutableDict*, *Resource*

Members permissions resource.

class sevenbridges.**Project**(***kwargs*)

Bases: *Resource*

Central resource for managing projects.

add_files(*files*)

Adds files to this project. :param files: List of files or a Collection object.

add_member(*user*, *permissions*)

Add a member to the project. :param user: Member username :param permissions: Permissions dictionary. :return: Member object.

add_member_division(*division*, *permissions*)

Add a member (team) to a project. :param division: Division object or division identifier. :param permissions: Permissions dictionary. :return: Member object.

add_member_email(*email*, *permissions=None*)

Add a member to the project using member email. :param email: Member email. :param permissions: Permissions dictionary. :return: Member object.

add_member_team(*team*, *permissions*)

Add a member (team) to a project. :param team: Team object or team identifier. :param permissions: Permissions dictionary. :return: Member object.

billing_group = None

category = None

classmethod create(*name*, *billing_group=None*, *description=None*, *tags=None*, *settings=None*, *api=None*)

Create a project. :param name: Project name. :param billing_group: Project billing group. :param description: Project description. :param tags: Project tags. :param settings: Project settings. :param api: Api instance. :return:

create_task(*name*, *app*, *revision=None*, *batch_input=None*, *batch_by=None*, *inputs=None*, *description=None*, *run=False*, *disable_batch=False*, *interruptible=True*, *execution_settings=None*)

Creates a task for this project.

Parameters

- **name** – Task name.
- **app** – CWL app identifier.
- **revision** – CWL app revision.
- **batch_input** – Batch input.
- **batch_by** – Batch criteria.
- **inputs** – Input map.
- **description** – Task description.
- **run** – True if you want to run a task upon creation.
- **disable_batch** – True if you want to disable batching.
- **interruptible** – True if you want to use interruptible instances.
- **execution_settings** – Execution settings for the task.

Returns
Task object.

created_by = None

created_on = None

deepcopy()

description = None

equals(*other*)

get_apps(*offset=None*, *limit=None*)

Retrieves apps in this project. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_exports(*volume=None*, *state=None*, *offset=None*, *limit=None*)

Fetches exports for this volume. :param volume: Optional volume identifier. :param state: Optional state. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_files(*offset=None*, *limit=None*)

Retrieves files in this project. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_imports(*volume=None*, *state=None*, *offset=None*, *limit=None*)

Fetches imports for this project. :param volume: Optional volume identifier. :param state: Optional state. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_member(*username*, *api=None*)

Fetches information about a single project member :param username: Member name :param api: Api instance :return: Member object

get_members(*offset=None*, *limit=None*)

Retrieves project members. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_tasks(*status=None*, *offset=None*, *limit=None*)

Retrieves tasks in this project. :param status: Optional task status. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

href = None

id = None

modified_on = None

name = None

classmethod query(*owner=None*, *name=None*, *offset=None*, *limit=None*, *api=None*, *category=None*, *tags=None*)

Query (List) projects :param owner: Owner username. :param name: Project name :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :param category: Project category. :param tags: Project tags. :return: Collection object.

remove_member(*user*)

Remove member from the project. :param user: User to be removed.

root_folder = None

save(**args*, ***kwargs*)

settings

tags = None

type = None

exception sevenbridges.**ReadOnlyPropertyError**(*message*)

Bases: *SbgError*

exception sevenbridges.**RequestTimeout**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.**ResourceNotModified**

Bases: *SbgError*

exception sevenbridges.**SbgError**(*message=None*, *code=None*, *status=None*, *more_info=None*)

Bases: Exception

Base class for SBG errors.

Provides a base exception for all errors that are thrown by sevenbridges-python library.

exception sevenbridges.**ServerError**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.**ServiceUnavailable**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

class sevenbridges.**Task**(***kwargs*)

Bases: *Resource*

Central resource for managing tasks.

abort(**args*, ***kwargs*)

app = None

batch = None

batch_by

batch_group

batch_input = None

classmethod bulk_get(*tasks*, *api=None*)

Retrieve tasks with specified ids in bulk :param tasks: Tasks to be retrieved. :param api: Api instance. :return: List of TaskBulkRecord objects.

clone(*run=True*)

Clone task :param run: run task after cloning :return: Task object.

classmethod create(*name*, *project*, *app*, *revision=None*, *batch_input=None*, *batch_by=None*, *inputs=None*, *description=None*, *run=False*, *disable_batch=False*, *interruptible=None*, *execution_settings=None*, *output_location=None*, *api=None*)

Creates a task on server. :param name: Task name. :param project: Project identifier. :param app: CWL app identifier. :param revision: CWL app revision. :param batch_input: Batch input. :param batch_by: Batch criteria. :param inputs: Input map. :param description: Task description. :param run: True if you want to run a task upon creation. :param disable_batch: If True disables batching of a batch task. :param interruptible: If True interruptible instance will be used. :param execution_settings: Execution settings for the task. :param output_location: Dictionary that allows you to define the exact location where your task outputs will be stored. :param api: Api instance. :return: Task object. :raises: TaskValidationError if validation Fails. :raises: SbgError if any exception occurs during request.

created_by = None

created_time = None

deepcopy()

description = None

end_time = None

equals(*other*)

errors = None

executed_by = None

execution_settings = None

execution_status

get_batch_children(*status=None*, *created_from=None*, *created_to=None*, *started_from=None*, *started_to=None*, *ended_from=None*, *ended_to=None*, *order_by=None*, *order=None*, *offset=None*, *limit=None*, *api=None*)

Retrieves batch child tasks for this task if its a batch task. :return: Collection instance. :raises SbError if task is not a batch task.

get_execution_details()

Retrieves execution details for a task. :return: Execution details instance.

href = None

id = None

inputs

name = None

origin = None

output_location = None

outputs

parent = None

price

project = None

classmethod query(*project=None*, *status=None*, *batch=None*, *parent=None*, *created_from=None*, *created_to=None*, *started_from=None*, *started_to=None*, *ended_from=None*, *ended_to=None*, *offset=None*, *limit=None*, *order_by=None*, *order=None*, *origin=None*, *api=None*)

Query (List) tasks. Date parameters may be both strings and python date objects. :param project: Target project. optional. :param status: Task status. :param batch: Only batch tasks. :param parent: Parent batch task identifier. :param ended_to: All tasks that ended until this date. :param ended_from: All tasks that ended from this date. :param started_to: All tasks that were started until this date. :param started_from: All tasks that were started from this date. :param created_to: All tasks that were created until this date. :param created_from: All tasks that were created from this date. :param offset: Pagination offset. :param limit: Pagination limit. :param order_by: Property to order by. :param order: Ascending or descending ordering. :param origin: Entity that created the task, e.g. automation run, if task was created by an automation run. :param api: Api instance. :return: Collection object.

run(**args*, ***kwargs*)

save(**args*, ***kwargs*)

start_time = None

status = None

type = None

use_interruptible_instances = None

wait(*period=10*, *callback=None*, **args*, ***kwargs*)

Wait until task is complete :param period: Time in seconds between reloads :param callback: Function to call after the task has finished, arguments and keyword arguments can be provided for it :return: Return value of provided callback function or None if a callback function was not provided

warnings = None

class sevenbridges.**TaskStatus**

Bases: `object`

ABORTED = 'ABORTED'

ABORTING = 'ABORTING'

COMPLETED = 'COMPLETED'

CREATING = 'CREATING'

DRAFT = 'DRAFT'

FAILED = 'FAILED'

QUEUED = 'QUEUED'

RUNNING = 'RUNNING'

terminal_states = ['COMPLETED', 'FAILED', 'ABORTED']

exception sevenbridges.**TaskValidationError**(*message*, *task=None*)

Bases: *SbgError*

class sevenbridges.**Team**(***kwargs*)

Bases: *Resource*

Central resource for managing teams.

add_member(*user*)

Add member to team :param user: User object or user's username :return: Added user.

classmethod create(*name*, *division*, *api=None*)

Create team within a division :param name: Team name. :param division: Parent division. :param api: Api instance. :return: Team object.

deepcopy()

equals(*other*)

get_members(*offset=None*, *limit=None*)

Fetch team members for current team. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

href = None

id = None

name = None

classmethod query(*division*, *list_all=False*, *offset=None*, *limit=None*, *api=None*)

Parameters

- **division** – Division slug.
- **list_all** – List all teams in division.
- **offset** – Pagination offset.
- **limit** – Pagination limit.
- **api** – Api instance.

Returns
Collection object.

remove_member(*user*)

Remove member from the team. :param user: User to be removed.

save(**args*, ***kwargs*)

class sevenbridges.**TeamMember**(***kwargs*)

Bases: *Resource*

Central resource for managing team members.

deepcopy()

equals(*other*)

href = None

id = None

role = None

username = None

exception sevenbridges.**TooManyRequests**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

class sevenbridges.**TransferState**

Bases: object

ABORTED = 'ABORTED'

COMPLETED = 'COMPLETED'

FAILED = 'FAILED'

PAUSED = 'PAUSED'

PREPARING = 'PREPARING'

RUNNING = 'RUNNING'

STOPPED = 'STOPPED'

exception sevenbridges.**Unauthorized**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

class sevenbridges.**User**(***kwargs*)

Bases: *Resource*

Central resource for managing users.

address = None

affiliation = None

city = None

country = None

deepcopy()

disable(*api=None*)

Disable user :param api: Api instance. :return:

email = None

equals(*other*)

first_name = None

classmethod get(*user*, *api=None*)

Fetches the resource from the server. :param id: Resource identifier :param api: sevenbridges Api instance. :return: Resource object.

href = None

last_name = None

classmethod me(*api=None*)

Retrieves current user information. :param api: Api instance. :return: User object.

phone = None

classmethod query(*division*, *role=None*, *offset=None*, *limit=None*, *api=None*)

Query division users :param division: Division slug. :param role: User role in division. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

role = None

state = None

username = None

zip_code = None

exception sevenbridges.**ValidationError**(*message*)

Bases: *SbgError*

class sevenbridges.**Volume**(***kwargs*)

Bases: *Resource*

Central resource for managing volumes.

access_mode = None

active = None

add_member(*user*, *permissions*)

Add a member to the volume. :param user: Member username :param permissions: Permissions dictionary. :return: Member object.

add_member_division(*division*, *permissions*)

Add a member (team) to a volume. :param division: Division object or division identifier. :param permissions: Permissions dictionary. :return: Member object.

add_member_team(*team*, *permissions*)

Add a member (team) to a volume. :param team: Team object or team identifier. :param permissions: Permissions dictionary. :return: Member object.

classmethod create_google_iam_volume(*name*, *bucket*, *configuration*, *access_mode*, *description=None*, *prefix=None*, *properties=None*, *api=None*)

Create google volume. :param name: Volume name. :param bucket: Referenced bucket. :param configuration: Google configuration. :param access_mode: Access Mode. :param description: Volume description. :param prefix: Volume prefix. :param properties: Volume properties. :param api: Api instance. :return: Volume object.

classmethod create_google_volume(*name*, *bucket*, *client_email*, *private_key*, *access_mode*, *description=None*, *prefix=None*, *properties=None*, *api=None*)

Create google volume. :param name: Volume name. :param bucket: Referenced bucket. :param client_email: Google client email. :param private_key: Google client private key. :param access_mode: Access Mode. :param description: Volume description. :param prefix: Volume prefix. :param properties: Volume properties. :param api: Api instance. :return: Volume object.

classmethod create_oss_volume(*name*, *bucket*, *endpoint*, *access_key_id*, *secret_access_key*, *access_mode*, *description=None*, *prefix=None*, *properties=None*, *api=None*)

Create oss volume. :param name: Volume name. :param bucket: Referenced bucket. :param access_key_id: Access key identifier. :param secret_access_key: Secret access key. :param access_mode:

Access Mode. :param endpoint: Volume Endpoint. :param description: Volume description. :param prefix: Volume prefix. :param properties: Volume properties. :param api: Api instance. :return: Volume object.

classmethod create_s3_volume(*name*, *bucket*, *access_key_id*, *secret_access_key*, *access_mode*, *description=None*, *prefix=None*, *properties=None*, *api=None*)

Create s3 volume. :param name: Volume name. :param bucket: Referenced bucket. :param access_key_id: Amazon access key identifier. :param secret_access_key: Amazon secret access key. :param access_mode: Access Mode. :param description: Volume description. :param prefix: Volume prefix. :param properties: Volume properties. :param api: Api instance. :return: Volume object.

classmethod create_s3_volume_role_auth(*name*, *bucket*, *role_arn*, *external_id*, *access_mode*, *description=None*, *prefix=None*, *properties=None*, *api=None*)

Create s3 volume using IAM Role auth. :param name: Volume name. :param bucket: Referenced bucket. :param role_arn: Amazon role ARN. :param external_id: Amazon role external id. :param access_mode: Access Mode. :param description: Volume description. :param prefix: Volume prefix. :param properties: Volume properties. :param api: Api instance. :return: Volume object.

created_on = None

deepcopy()

description = None

equals(*other*)

get_exports(*state=None*, *offset=None*, *limit=None*)

Fetches exports for this volume. :param state: Optional state. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_imports(*project=None*, *state=None*, *offset=None*, *limit=None*)

Fetches imports for this volume. :param project: Optional project identifier. :param state: Optional state. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_member(*username*, *api=None*)

Fetches information about a single volume member :param username: Member name :param api: Api instance :return: Member object

get_members(*offset=None*, *limit=None*)

Retrieves volume members. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_volume_object_info(*location*)

Fetches information about single volume object - usually file :param location: object location :return:

href = None

id = None

list(*prefix=None*, *limit=None*, *fields='_all'*)

modified_on = None

name = None

classmethod query(*offset=None*, *limit=None*, *api=None*)

Query (List) volumes. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

remove_member(*user*)

Remove member from the volume. :param user: User to be removed.

save(**args*, ***kwargs*)

service

class sevenbridges.**VolumeAccessMode**

Bases: object

READ_ONLY = 'RO'

READ_WRITE = 'RW'

class sevenbridges.**VolumeObject**(***kwargs*)

Bases: *Resource*

Volume object resource contains information about single file (object) entry in a specific volume.

deepcopy()

equals(*other*)

href = None

location = None

metadata = None

type = None

volume = None

class sevenbridges.**VolumeType**

Bases: object

GOOGLE = 'GCS'

OSS = 'OSS'

S3 = 'S3'

1.4.2 Subpackages

sevenbridges.http package

Submodules

sevenbridges.http.client module

class sevenbridges.http.client.**AAHeader**

Bases: object

key = 'X-Sbg-Advance-Access'

value = 'Advance'

class sevenbridges.http.client.**HttpClient**(*url=None*, *token=None*, *oauth_token=None*, *config=None*, *timeout=None*, *proxies=None*, *error_handlers=None*, *advance_access=False*, *pool_connections=None*, *pool_maxsize=None*, *pool_block=True*, *max_parallel_requests=None*, *retry_count=None*, *backoff_factor=None*)

Bases: object

Implementation of all low-level API stuff, creating and sending requests, returning raw responses, authorization, etc.

add_error_handler(*handler*)

delete(*url*, *headers=None*, *params=None*, *append_base=True*)

get(*url*, *headers=None*, *params=None*, *data=None*, *append_base=True*, *stream=False*)

property limit

patch(*url*, *headers=None*, *params=None*, *data=None*, *append_base=True*)

post(*url*, *headers=None*, *params=None*, *data=None*, *append_base=True*)

put(*url*, *headers=None*, *params=None*, *data=None*, *append_base=True*)

property remaining

remove_error_handler(*handler*)

property request_id

property reset_time

property session

class sevenbridges.http.client.**RequestSession**

Bases: Session

Client session class

send(*request*, ***kwargs*)

Send prepared request :param request: Prepared request to be sent :param kwargs: request keyword arguments :return: Request response

sevenbridges.http.client.**config_vars**(*profiles*, *advance_access*)

Utility method to fetch config vars using ini section profile :param profiles: profile name. :param advance_access: advance_access flag. :return:

sevenbridges.http.client.**generate_session**(*pool_connections*, *pool_maxsize*, *pool_block*, *proxies=None*, *retry_count=None*, *backoff_factor=None*)

Utility method to generate request sessions. :param pool_connections: The number of urllib3 connection pools to

cache.

Parameters

- **pool_maxsize** – The maximum number of connections to save in the pool.
- **pool_block** – Whether the connection pool should block for connections.
- **proxies** – Proxies dictionary.
- **retry_count** – Number of retries to attempt
- **backoff_factor** – Backoff factor for retries

Returns
requests.Session object.

sevenbridges.http.client.**mask_secrets**(*request_data*)

sevenbridges.http.error_handlers module

sevenbridges.http.error_handlers.**general_error_sleeper**(*api*, *response*, *sleep=300*)

Pauses the execution if response status code is > 500. :param api: Api instance. :param response: requests.Response object :param sleep: Time to sleep in between the requests.

sevenbridges.http.error_handlers.**maintenance_sleeper**(*api*, *response*, *sleep=300*)

Pauses the execution if sevenbridges api is under maintenance. :param api: Api instance. :param response: requests.Response object. :param sleep: Time to sleep in between the requests.

sevenbridges.http.error_handlers.**rate_limit_sleeper**(*api*, *response*)

Pauses the execution if rate limit is breached. :param api: Api instance. :param response: requests.Response object

sevenbridges.http.error_handlers.**repeatable_handler**(*f*)

Marks 'repeatable' error handlers. Error handler is repeatable if propagate input response in case of no error handling occurred.

sevenbridges.meta package

Submodules

sevenbridges.meta.collection module

class sevenbridges.meta.collection.**Collection**(*resource*, *href*, *total*, *items*, *links*, *api*)

Bases: `list`

Wrapper for SevenBridges pageable resources. Among the actual collection items it contains information regarding the total number of entries available in on the server and resource href.

all()

Fetches all available items. :return: Collection object.

next_page()

Fetches next result set. :return: Collection object.

previous_page()

Fetches previous result set. :return: Collection object.

resource = None

property total

class sevenbridges.meta.collection.**VolumeCollection**(*href*, *items*, *links*, *prefixes*, *api*)

Bases: *Collection*

next_page()

Fetches next result set. :return: VolumeCollection object.

previous_page()

Fetches previous result set. :return: Collection object.

property total

sevenbridges.meta.comp_mutable_dict module

class sevenbridges.meta.comp_mutable_dict.**CompoundMutableDict**(***kwargs*)

Bases: dict

Resource used for mutable compound dictionaries.

equals(*other*)

items() → a set-like object providing a view on D's items

update([*E*], ***F*) → None. Update D from dict/iterable E and F.

If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

sevenbridges.meta.data module

class sevenbridges.meta.data.**DataContainer**(*urls*, *api*, *parent*)

Bases: object

Utility for fetching data from the API server using, resource identifier or href.

fetch(*item=None*)

sevenbridges.meta.fields module

class sevenbridges.meta.fields.**BasicListField**(*read_only*, *name=None*, *max_length=None*)

Bases: *Field*

validate(*value*)

class sevenbridges.meta.fields.**BooleanField**(*read_only*, *name=None*)

Bases: *Field*

validate(*value*)

class sevenbridges.meta.fields.**CompoundField**(*cls*, *read_only*, *name=None*, *validator=None*)

Bases: *Field*

class sevenbridges.meta.fields.**CompoundListField**(*cls*, *read_only*, *name=None*)

Bases: *Field*

class sevenbridges.meta.fields.**DateTimeField**(*read_only*, *name=None*)

Bases: *Field*

class sevenbridges.meta.fields.**DictField**(*read_only*, *name=None*)

Bases: *Field*, dict

class sevenbridges.meta.fields.**Field**(*name=None*, *read_only=True*, *validator=None*)

Bases: object

EMPTY = <object object>

validate(*value*)

class sevenbridges.meta.fields.**FloatField**(*read_only*, *name=None*)

Bases: *Field*

validate(*value*)

class sevenbridges.meta.fields.**HrefField**(*read_only*, *name=None*)

Bases: *Field*

class sevenbridges.meta.fields.**IntegerField**(*read_only*, *name=None*)

Bases: *Field*

validate(*value*)

class sevenbridges.meta.fields.**ObjectIdField**(*read_only*, *name=None*)

Bases: *Field*

class sevenbridges.meta.fields.**StringField**(*read_only*, *name=None*, *max_length=None*)

Bases: *Field*

validate(*value*)

class sevenbridges.meta.fields.**UuidField**(*read_only*, *name=None*)

Bases: *Field*

validate(*value*)

sevenbridges.meta.resource module

class sevenbridges.meta.resource.**Resource**(*api*, **args*, ***kwargs*)

Bases: object

Resource is base class for all resources, hiding implementation details of magic of injecting instance of API and common operations (like generic query).

delete()

Deletes the resource on the server.

field(*name*)

Return field value if it's set :param name: Field name :return: Field value or None

classmethod get(*id*, *api=None*)

Fetches the resource from the server. :param id: Resource identifier :param api: sevenbridges Api instance. :return: Resource object.

reload()

Refreshes the resource with the data from the server.

update_old()

class sevenbridges.meta.resource.**ResourceMeta**(*name*, *bases*, *dct*)

Bases: type

Metaclass for all resources, knows how to inject instance of API from class that contains classes with this meta. Class that contains this class has to have 'api' property which will be injected into class level API property of Resource class.

Creates constructors for all resources and manages instantiation of resource fields.

sevenbridges.meta.transformer module

class sevenbridges.meta.transformer.**Transform**

Bases: object

static to_app(*app*)

Serializes app to id string :param app: object to serialize :return: string id

static to_async_job(*async_job*)

static to_automation(*automation*)

static to_automation_member(*member*)

static to_automation_package(*package*)

static to_automation_run(*automation_run*)

static to_billing_group(*billing_group*)

Serializes billing_group to id string :param billing_group: object to serialize :return: string id

static to_dataset(*dataset*)

static to_datestring(*d*)

Serializes date to string :param d: object to serialize :return: string date

static to_division(*division*)

Serializes division to id string :param division: object to serialize :return: string id

static to_export(*export*)

Serializes export to id string :param export: object to serialize :return: string id

static to_file(*file_*)

Serializes file to id string :param **file_**: object to serialize :return: string id

static to_import(*import_*)

Serializes import to id string :param **import_**: object to serialize :return: string id

static to_location(*location*)

Serializes location to string :param location: object to serialize :return: string

static to_marker(*marker*)

Serializes marker to string :param marker: object to serialize :return: string id

static to_member(*member*)

static to_project(*project*)

Serializes project to id string :param project: object to serialize :return: string id

static to_resource(*resource*)

static to_tags(*tags*)

static to_task(*task*)

Serializes task to id string :param task: object to serialize :return: string id

static to_team(*team*)

Serializes team to id string :param team: object to serialize :return: string id

static to_user(*user*)

Serializes user to id string :param user: object to serialize :return: string id

static to_volume(*volume*)

Serializes volume to id string :param volume: object to serialize :return: string id

sevenbridges.models package

Subpackages

sevenbridges.models.compound package

Subpackages

sevenbridges.models.compound.billing package

Submodules

sevenbridges.models.compound.billing.invoice_period module

class sevenbridges.models.compound.billing.invoice_period.**InvoicePeriod**(***kwargs*)

Bases: *Resource*

Invoice period resource contains datetime information about the invoice. It has from and to fields which represent the interval period for this invoice.

deepcopy()

equals(*other*)

from_ = None

to = None

sevenbridges.models.compound.billing.project_breakdown module

class sevenbridges.models.compound.billing.project_breakdown.**ProjectBreakdown**(***kwargs*)

Bases: *Resource*

Project breakdown resource contains information regarding billing group project breakdown costs.

analysis_spending

deepcopy()

equals(*other*)

href = None

task_breakdown

sevenbridges.models.compound.billing.task_breakdown module

class sevenbridges.models.compound.billing.task_breakdown.**TaskBreakdown**(***kwargs*)

Bases: *Resource*

Task breakdown resource contains information regarding billing group analysis breakdown costs.

deepcopy()

equals(*other*)

href = None

runner_username = None

task_cost

time_finished = None

time_started = None

sevenbridges.models.compound.files package

Submodules

sevenbridges.models.compound.files.download_info module

class sevenbridges.models.compound.files.download_info.**DownloadInfo**(***kwargs*)

Bases: *Resource*

Download info resource contains download url for the file.

deepcopy()

equals(*other*)

url = None

sevenbridges.models.compound.files.file_origin module

class sevenbridges.models.compound.files.file_origin.**FileOrigin**(***kwargs*)

Bases: *Resource*

File origin resource contains information about origin of a file. Among others it contains information about the task if the file was produced during executions of a analysis.

deepcopy()

equals(*other*)

task = None

sevenbridges.models.compound.files.file_storage module

class sevenbridges.models.compound.files.file_storage.**FileStorage**(***kwargs*)

Bases: *Resource*

File storage resource contains information about the storage location of the file if the file is imported on or exported to an external volume.

deepcopy()

equals(*other*)

location = None

type = None

volume = None

sevenbridges.models.compound.files.metadata module

class sevenbridges.models.compound.files.metadata.**Metadata**(***kwargs*)

Bases: *CompoundMutableDict*, *Resource*

File metadata resource.

sevenbridges.models.compound.jobs package

Submodules

sevenbridges.models.compound.jobs.job module

class sevenbridges.models.compound.jobs.job.**Job**(***kwargs*)

Bases: *Resource*

Job resource contains information for a single executed node in the analysis.

command_line = None

deepcopy()

docker

end_time = None

equals(*other*)

instance

logs

name = None

retried = None

start_time = None

status = None

sevenbridges.models.compound.jobs.job_docker module

class sevenbridges.models.compound.jobs.job_docker.**JobDocker**(***kwargs*)

Bases: *Resource*

JobDocker resource contains information for a docker image that was used for execution of a single job.

checksum = None

deepcopy()

equals(*other*)

sevenbridges.models.compound.jobs.job_instance module

class sevenbridges.models.compound.jobs.job_instance.**Instance**(***kwargs*)

Bases: *Resource*

Instance resource contains information regarding the instance on which the job was executed.

deepcopy()

disk

equals(*other*)

id = None

provider = None

type = None

sevenbridges.models.compound.jobs.job_instance_disk module

class sevenbridges.models.compound.jobs.job_instance_disk.**Disk**(***kwargs*)

Bases: *Resource*

Disk resource contains information about EBS disk size.

deepcopy()

equals(*other*)

size = None

type = None

unit = None

sevenbridges.models.compound.jobs.job_log module

class sevenbridges.models.compound.jobs.job_log.**Logs**(***kwargs*)

Bases: *CompoundMutableDict*, *Resource*

Task output resource.

sevenbridges.models.compound.limits package

Submodules

sevenbridges.models.compound.limits.rate module

class sevenbridges.models.compound.limits.rate.**Rate**(***kwargs*)

Bases: *Resource*

Rate resource.

deepcopy()

equals(*other*)

limit = None

remaining = None

reset = None

sevenbridges.models.compound.markers package

Submodules

sevenbridges.models.compound.markers.position module

class sevenbridges.models.compound.markers.position.**MarkerPosition**(***kwargs*)

Bases: *CompoundMutableDict*, *Resource*

Marker position resource

sevenbridges.models.compound.projects package

Submodules

sevenbridges.models.compound.projects.permissions module

class sevenbridges.models.compound.projects.permissions.**Permissions**(***kwargs*)

Bases: *CompoundMutableDict*, *Resource*

Members permissions resource.

sevenbridges.models.compound.projects.settings module

class sevenbridges.models.compound.projects.settings.**Settings**(***kwargs*)

Bases: *CompoundMutableDict*, *Resource*

Project settings resource.

sevenbridges.models.compound.tasks package

sevenbridges.models.compound.tasks.**map_input_output**(*item*, *api*)

Maps item to appropriate sevebridges object. :param item: Input/Output value. :param api: Api instance. :return: Mapped object.

Submodules

sevenbridges.models.compound.tasks.batch_by module

class sevenbridges.models.compound.tasks.batch_by.**BatchBy**(***kwargs*)

Bases: *Resource*, `dict`

Task batch by resource.

equals(*other*)

update([*E*], ***F*) → None. Update D from dict/iterable E and F.

If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

sevenbridges.models.compound.tasks.batch_group module

class sevenbridges.models.compound.tasks.batch_group.**BatchGroup**(***kwargs*)

Bases: *Resource*

Batch group for a batch task. Represents the group that is assigned to the child task from the batching criteria that was used when the task was started.

deepcopy()

equals(*other*)

fields = None

value = None

sevenbridges.models.compound.tasks.execution_status module

class sevenbridges.models.compound.tasks.execution_status.**ExecutionStatus**(***kwargs*)

Bases: *Resource*

Task execution status resource.

Contains information about the number of completed task steps, total number of task steps, current execution message and information regarding computation limits.

In case of a batch task it also contains the number of queued, running, completed, failed and aborted tasks.

aborted = None

account_limit = None

completed = None

deepcopy()

duration = None

equals(*other*)

execution_duration = None

failed = None

instance_init = None

message = None

message_code = None

queued = None

queued_duration = None

running = None

running_duration = None

steps_completed = None

steps_total = None

system_limit = None

sevenbridges.models.compound.tasks.input module

class sevenbridges.models.compound.tasks.input.**Input**(***kwargs*)

Bases: *CompoundMutableDict*, *Resource*

Task input resource.

sevenbridges.models.compound.tasks.output module

class sevenbridges.models.compound.tasks.output.**Output**(***kwargs*)

Bases: *CompoundMutableDict*, *Resource*

Task output resource.

sevenbridges.models.compound.volumes package

Submodules

sevenbridges.models.compound.volumes.import_destination module

class sevenbridges.models.compound.volumes.import_destination.**ImportDestination**(***kwargs*)

Bases: *Resource*

ImportDestination resource describes the location of the file imported on to SevenBridges platform or related product.

deepcopy()

equals(*other*)

name = None

parent = None

project = None

sevenbridges.models.compound.volumes.properties module

class sevenbridges.models.compound.volumes.properties.**VolumeProperties**(***kwargs*)

Bases: *CompoundMutableDict*, *Resource*

Volume permissions resource.

sevenbridges.models.compound.volumes.service module

class sevenbridges.models.compound.volumes.service.**VolumeService**(***kwargs*)

Bases: *CompoundMutableDict*, *Resource*

Volume Service resource. Contains information about external storage provider.

sevenbridges.models.compound.volumes.volume_file module

class sevenbridges.models.compound.volumes.volume_file.**VolumeFile**(***kwargs*)

Bases: *Resource*

VolumeFile resource describes the location of the file on the external volume.

deepcopy()

equals(*other*)

location = None

volume = None

sevenbridges.models.compound.volumes.volume_object module

class sevenbridges.models.compound.volumes.volume_object.**VolumeObject**(***kwargs*)

Bases: *Resource*

Volume object resource contains information about single file (object) entry in a specific volume.

deepcopy()

equals(*other*)

href = None

location = None

metadata = None

type = None

volume = None

sevenbridges.models.compound.volumes.volume_prefix module

class sevenbridges.models.compound.volumes.volume_prefix.**VolumePrefix**(***kwargs*)

Bases: *Resource*

Volume prefix resource contains information about volume prefixes

deepcopy()

equals(*other*)

href = None

prefix = None

volume = None

Submodules

sevenbridges.models.compound.error module

class sevenbridges.models.compound.error.**Error**(***kwargs*)

Bases: *Resource*

Error resource describes the error that happened and provides http status, custom codes and messages as well as the link to online resources.

code = None

deepcopy()

equals(*other*)

message = None

more_info = None

status = None

sevenbridges.models.compound.price module

class sevenbridges.models.compound.price.**Price**(***kwargs*)

Bases: *Resource*

Price resource contains an information regarding the currency and the monet value of a certain resource.

amount = None

breakdown

currency = None

deepcopy()

equals(*other*)

sevenbridges.models.compound.price_breakdown module

class sevenbridges.models.compound.price_breakdown.**Breakdown**(***kwargs*)

Bases: *Resource*

Breakdown resource contains price breakdown by storage and computation.

computation = None

data_transfer = None

deepcopy()

equals(*other*)

storage = None

Submodules

sevenbridges.models.actions module

class sevenbridges.models.actions.**Actions**(***kwargs*)

Bases: *Resource*

classmethod bulk_copy_files(*files*, *destination_project*, *api=None*)

Bulk copy of files. :param files: List containing files to be copied. :param destination_project: Destination project. :param api: Api instance. :return: MultiStatus copy result.

deepcopy()

equals(*other*)

classmethod send_feedback(*type='IDEA'*, *referrer=None*, *text=None*, *api=None*)

Sends feedback to sevenbridges. :param type: FeedbackType wither IDEA, PROBLEM or THOUGHT. :param text: Feedback text. :param referrer: Feedback referrer. :param api: Api instance.

sevenbridges.models.app module

class sevenbridges.models.app.**App**(***kwargs*)

Bases: *Resource*

Central resource for managing apps.

copy(*project*, *name=None*, *strategy=None*, *use_revision=False*, *api=None*)

Copies the current app. :param project: Destination project. :param name: Destination app name. :param strategy: App copy strategy. :param use_revision: Copy from set app revision. :param api: Api instance. :return: Copied App object.

Copy strategies

clone copy all revisions and continue getting updates form the original app (default method when the key is omitted)

direct copy only the latest revision and get the updates from this point on

clone_direct copy the app like the direct strategy, but keep all revisions

transient copy only the latest revision and continue getting updates from the original app

classmethod create_revision(*id*, *revision*, *raw*, *api=None*)

Create a new app revision. :param id: App identifier. :param revision: App revision. :param raw: Raw cwl object. :param api: Api instance. :return: App object.

deepcopy()

equals(*other*)

classmethod get_revision(*id*, *revision*, *api=None*)

Get app revision. :param id: App identifier. :param revision: App revision :param api: Api instance. :return: App object.

href = None

property id

classmethod install_app(*id*, *raw*, *api=None*, *raw_format=None*)

Installs and app. :param id: App identifier. :param raw: Raw cwl data. :param api: Api instance. :param raw_format: Format of raw app data being sent, json by default :return: App object.

name = None

project = None

classmethod query(*project=None*, *visibility=None*, *q=None*, *id=None*, *offset=None*, *limit=None*, *api=None*)

Query (List) apps. :param project: Source project. :param visibility: private|public for private or public apps. :param q: List containing search terms. :param id: List contains app ids. Fetch apps with specific ids. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: collection object

raw = None

revision = None

sync()

Syncs the parent app changes with the current app instance. :return: Synced App object.

sevenbridges.models.billing_egress_breakdown module

class sevenbridges.models.billing_egress_breakdown.**BillingGroupEgressBreakdown**(***kwargs*)

Bases: *Resource*

deepcopy()

downloaded

downloaded_by = None

egress_cost

equals(*other*)

project_locked = None

project_name = None

classmethod query(*bg_id*, *api=None*, *date_from=None*, *date_to=None*, *invoice_id=None*, *fields=None*, *offset=None*, *limit=None*)

Query (List) billing group egress breakdown. Date parameters must be string in format MM-DD-YYYY

Parameters

- **fields** –
- **invoice_id** –
- **date_to** – include all egress transactions charged before and including date_to
- **date_from** – include all egress transactions charged after and including date_from
- **bg_id** – Billing Group ID
- **offset** – Pagination offset.
- **limit** – Pagination limit.
- **api** – Api instance.

Returns
Collection object.

sevenbridges.models.billing_storage_breakdown module

class sevenbridges.models.billing_storage_breakdown.**BillingGroupStorageBreakdown**(***kwargs*)

Bases: *Resource*

active

archived

deepcopy()

equals(*other*)

location = None

project_created_by = None

project_locked = None

project_name = None

classmethod query(*bg_id*, *api=None*, *date_from=None*, *date_to=None*, *invoice_id=None*, *fields=None*, *offset=None*, *limit=None*)

Query (List) billing group storage breakdown. Date parameters must be string in format MM-DD-YYYY

Parameters

- **fields** –
- **invoice_id** –
- **date_to** – include all storage transactions charged before and including date_to
- **date_from** – include all storage transactions charged after and including date_from
- **bg_id** – Billing Group ID
- **offset** – Pagination offset.
- **limit** – Pagination limit.
- **api** – Api instance.

Returns
Collection object.

sevenbridges.models.billing_group module

class sevenbridges.models.billing_group.**BillingGroup**(***kwargs*)

Bases: *Resource*

Central resource for managing billing groups.

analysis_breakdown(*date_from=None*, *date_to=None*, *invoice_id=None*, *fields=None*, *offset=None*, *limit=None*)

Get Billing group analysis breakdown for the current billing group.

balance

deepcopy()

disabled = None

egress_breakdown(*date_from=None*, *date_to=None*, *invoice_id=None*, *fields=None*, *offset=None*, *limit=None*)

Get Billing group egress breakdown for the current billing group.

equals(*other*)

href = None

id = None

name = None

owner = None

pending = None

classmethod query(*offset=None*, *limit=None*, *api=None*)

Query (List) billing group. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object. :param api: Api instance.

storage_breakdown(*date_from=None*, *date_to=None*, *invoice_id=None*, *fields=None*, *offset=None*, *limit=None*)

Get Billing group storage breakdown for the current billing group.

type = None

sevenbridges.models.division module

class sevenbridges.models.division.**Division**(***kwargs*)

Bases: *Resource*

Central resource for managing divisions.

deepcopy()

equals(*other*)

get_members(*role=None*, *offset=None*, *limit=None*)

get_teams(*offset=None*, *limit=None*)

href = None

id = None

name = None

classmethod query(*offset=None*, *limit=None*, *api=None*)

Query (List) divisions.

Parameters

- **offset** – Pagination offset.
- **limit** – Pagination limit.
- **api** – Api instance.

Returns

Collection object.

sevenbridges.models.endpoints module

class sevenbridges.models.endpoints.**Endpoints**(***kwargs*)

Bases: *Resource*

Central resource for managing Endpoints.

action_url = None

apps_url = None

billing_url = None

deepcopy()

equals(*other*)

files_url = None

classmethod get(*api=None*, ***kwargs*)

Get api links. :param api: Api instance. :return: Endpoints object.

projects_url = None

rate_limit_url = None

tasks_url = None

upload_url = None

user_url = None

users_url = None

sevenbridges.models.enums module

class sevenbridges.models.enums.**AppCopyStrategy**

Bases: object

CLONE = 'clone'

CLONE_DIRECT = 'clone_direct'

DIRECT = 'direct'

TRANSIENT = 'transient'

class sevenbridges.models.enums.**AppRawFormat**

Bases: object

JSON = 'json'

YAML = 'yaml'

class sevenbridges.models.enums.**AsyncFileOperations**

Bases: object

COPY = 'copy'

DELETE = 'delete'

MOVE = 'move'

class sevenbridges.models.enums.**AsyncJobStates**

Bases: object

FINISHED = 'FINISHED'

RESOLVING = 'RESOLVING'

RUNNING = 'RUNNING'

SUBMITTED = 'SUBMITTED'

class sevenbridges.models.enums.**AutomationRunActions**

Bases: object

RERUN = 'rerun'

STOP = 'stop'

class sevenbridges.models.enums.**AutomationStatus**

Bases: object

ABORTED = 'ABORTED'

CREATED = 'CREATED'

FAILED = 'FAILED'

FINISHED = 'FINISHED'

QUEUED_FOR_EXECUTION = 'QUEUED_FOR_EXECUTION'

QUEUED_FOR_TERMINATION = 'QUEUED_FOR_TERMINATION'

RUNNING = 'RUNNING'

SENT_TO_EXECUTION = 'SENT_TO_EXECUTION'

terminal_states = ['FINISHED', 'FAILED', 'ABORTED']

class sevenbridges.models.enums.DivisionRole

Bases: object

ADMIN = 'admin'

EXTERNAL_COLLABORATOR = 'external_collaborator'

MEMBER = 'member'

class sevenbridges.models.enums.FeedbackType

Bases: object

IDEA = 'IDEA'

PROBLEM = 'PROBLEM'

THOUGHT = 'THOUGHT'

class sevenbridges.models.enums.FileApiFormats

Bases: object

FILE = 'File'

FOLDER = 'Directory'

class sevenbridges.models.enums.FileStorageType

Bases: object

PLATFORM = 'PLATFORM'

VOLUME = 'VOLUME'

class sevenbridges.models.enums.ImportExportState

Bases: object

COMPLETED = 'COMPLETED'

FAILED = 'FAILED'

PENDING = 'PENDING'

RUNNING = 'RUNNING'

class sevenbridges.models.enums.PartSize

Bases: object

DOWNLOAD_MINIMUM_PART_SIZE = 5242880

GB = 1073741824

KB = 1024

MAXIMUM_OBJECT_SIZE = 5497558138880

MAXIMUM_TOTAL_PARTS = 10000

MAXIMUM_UPLOAD_SIZE = 5368709120

MB = 1048576

TB = 1099511627776

UPLOAD_MINIMUM_PART_SIZE = 5242880

UPLOAD_RECOMMENDED_SIZE = 33554432

class sevenbridges.models.enums.RequestParameters

Bases: object

DEFAULT_BACKOFF_FACTOR = 1

DEFAULT_BULK_LIMIT = 100

DEFAULT_RETRY_COUNT = 6

DEFAULT_TIMEOUT = 300

MAX_URL_LENGTH = 6000

class sevenbridges.models.enums.TaskStatus

Bases: object

ABORTED = 'ABORTED'

ABORTING = 'ABORTING'

COMPLETED = 'COMPLETED'

CREATING = 'CREATING'

DRAFT = 'DRAFT'

FAILED = 'FAILED'

QUEUED = 'QUEUED'

RUNNING = 'RUNNING'

terminal_states = ['COMPLETED', 'FAILED', 'ABORTED']

class sevenbridges.models.enums.TransferState

Bases: object

ABORTED = 'ABORTED'

COMPLETED = 'COMPLETED'

FAILED = 'FAILED'

PAUSED = 'PAUSED'

PREPARING = 'PREPARING'

RUNNING = 'RUNNING'

STOPPED = 'STOPPED'

class sevenbridges.models.enums.VolumeAccessMode

Bases: object

READ_ONLY = 'RO'

READ_WRITE = 'RW'

class sevenbridges.models.enums.VolumeType

Bases: object

GOOGLE = 'GCS'

OSS = 'OSS'

S3 = 'S3'

sevenbridges.models.execution_details module

class sevenbridges.models.execution_details.ExecutionDetails(**kwargs)

Bases: *Resource*

Task execution details.

deepcopy()

end_time = None

equals(*other*)

href = None

jobs

message = None

start_time = None

status = None

sevenbridges.models.file module

class sevenbridges.models.file.File(**kwargs)

Bases: *Resource*

Central resource for managing files.

FOLDER_TYPE = 'folder'

classmethod bulk_delete(*files*, *api=None*)

Delete files with specified ids in bulk :param files: Files to be deleted. :param api: Api instance. :return: List of FileBulkRecord objects.

classmethod bulk_edit(*files*, *api=None*)

This call edits the details for multiple specified files. Use this call to modify the existing information for the files or add new information while preserving omitted parameters. For each of the specified files, the call edits its name, tags and metadata. :param files: List of file instances. :param api: Api instance. :return: List of FileBulkRecord objects.

classmethod bulk_get(*files*, *api=None*)

Retrieve files with specified ids in bulk :param files: Files to be retrieved. :param api: Api instance. :return: List of FileBulkRecord objects.

classmethod bulk_update(*files*, *api=None*)

This call updates the details for multiple specified files. Use this call to set new information for the files, thus replacing all existing information and erasing omitted parameters. For each of the specified files, the call sets a new name, new tags and metadata. :param files: List of file instances. :param api: Api instance. :return: List of FileBulkRecord objects.

content(*path=None*, *overwrite=True*, *encoding='utf-8'*)

Downloads file to the specified path or as temporary file and reads the file content in memory. Should not be used on very large files. :param path: Path for file download If omitted tmp file will be used. :param overwrite: Overwrite file if exists locally :param encoding: File encoding, by default it is UTF-8 :return: File content.

copy(*project*, *name=None*)

Copies the current file. :param project: Destination project. :param name: Destination file name. :return: Copied File object.

copy_to_folder(*parent*, *name=None*, *api=None*)

Copy file to folder :param parent: Folder to copy file to :param name: New file name :param api: Api instance :return: New file instance

classmethod create_folder(*name*, *parent=None*, *project=None*, *api=None*)

Create a new folder :param name: Folder name :param parent: Parent folder :param project: Project to create folder in :param api: Api instance :return: New folder

created_on = None

deepcopy()

download(*path*, *retry=6*, *timeout=300*, *chunk_size=None*, *wait=True*, *overwrite=False*)

Downloads the file and returns a download handle. Download will not start until .start() method is invoked. :param path: Full path to the new file. :param retry: Number of retries if error occurs during download. :param timeout: Timeout for http requests. :param chunk_size: Chunk size in bytes. :param wait: If true will wait for download to complete. :param overwrite: If True will silently overwrite existing file. :return: Download handle.

download_info()

Fetches download information containing file url that can be used to download file. :return: Download info object.

equals(*other*)

href = None

id = None

is_folder()

list_files(*offset=None*, *limit=None*, *api=None*, *cont_token=None*)

List files in a folder :param api: Api instance :param offset: Pagination offset :param limit: Pagination limit :param cont_token: Pagination continuation token :return: List of files

metadata

modified_on = None

move_to_folder(*parent*, *name=None*, *api=None*)

Move file to folder :param parent: Folder to move file to :param name: New file name :param api: Api instance :return: New file instance

name = None

origin

parent = None

project = None

classmethod query(*project=None*, *names=None*, *metadata=None*, *origin=None*, *tags=None*, *offset=None*, *limit=None*, *dataset=None*, *api=None*, *parent=None*, *cont_token=None*)

Query (List) files, requires project or dataset :param project: Project id :param names: Name list :param metadata: Metadata query dict :param origin: Origin query dict :param tags: List of tags to filter on :param offset: Pagination offset :param limit: Pagination limit :param dataset: Dataset id :param api: Api instance. :param parent: Folder id or File object with type folder :param cont_token: Pagination continuation token :return: Collection object.

reload()

Refreshes the file with the data from the server.

save(**args*, ***kwargs*)

classmethod search(*query*, *cont_token=None*, *limit=None*, *api=None*)

Search files by a query. :param query: Query written in SBG query language. :param cont_token: Continuation token value. :param limit: Limit value. :param api: Api instance.

property secondary_files

size = None

storage

stream(*part_size=32768*)

Creates an iterator which can be used to stream the file content. :param part_size: Size of the part in bytes. Default 32KB :return Iterator

tags = None

type = None

classmethod upload(*path*, *project=None*, *parent=None*, *file_name=None*, *overwrite=False*, *retry=6*, *timeout=300*, *part_size=None*, *wait=True*, *api=None*)

Uploads a file using multipart upload and returns an upload handle if the wait parameter is set to False. If wait is set to True it will block until the upload is completed.

Parameters

- **path** – File path on local disc.

- **project** – Project identifier
- **parent** – Parent folder identifier
- **file_name** – Optional file name.
- **overwrite** – If true will overwrite the file on the server.
- **retry** – Number of retries if error occurs during upload.
- **timeout** – Timeout for http requests.
- **part_size** – Part size in bytes.
- **wait** – If true will wait for upload to complete.
- **api** – Api instance.

class sevenbridges.models.file.**FileBulkRecord**(***kwargs*)

Bases: *BulkRecord*

deepcopy()

equals(*other*)

resource

class sevenbridges.models.file.**SearchResponse**(***kwargs*)

Bases: *Resource*

cont_token = None

count = None

deepcopy()

equals(*other*)

result_set = None

sevenbridges.models.invoice module

class sevenbridges.models.invoice.**Invoice**(***kwargs*)

Bases: *Resource*

Central resource for managing invoices.

analysis_costs

deepcopy()

equals(*other*)

href = None

id = None

invoice_period

pending = None

classmethod query(*offset=None*, *limit=None*, *api=None*)

Query (List) invoices. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

storage_costs

total

sevenbridges.models.link module

class sevenbridges.models.link.**Link**(***kwargs*)

Bases: *Resource*

Pagination links.

deepcopy()

equals(*other*)

href = None

method = None

rel = None

class sevenbridges.models.link.**VolumeLink**(***kwargs*)

Bases: *Resource*

Pagination links for volumes.

deepcopy()

equals(*other*)

next = None

sevenbridges.models.marker module

class sevenbridges.models.marker.**Marker**(***kwargs*)

Bases: *Resource*

chromosome = None

classmethod create(*file*, *name*, *position*, *chromosome*, *private=True*, *api=None*)

Create a marker on a file. :param file: File object or identifier. :param name: Marker name. :param position: Marker position object. :param chromosome: Chromosome number. :param private: Whether the marker is private or public. :param api: Api instance. :return: Marker object.

created_by = None

created_time = None

deepcopy()

equals(*other*)

file = None

href = None

id = None

name = None

position

classmethod query(*file*, *offset=None*, *limit=None*, *api=None*)

Queries genome markers on a file. :param file: Genome file - Usually bam file. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

save(**args*, ***kwargs*)

sevenbridges.models.member module

class sevenbridges.models.member.**Member**(***kwargs*)

Bases: *Resource*

Central resource for managing members. This resource is reused on both projects and volumes.

deepcopy()

email = None

equals(*other*)

href = None

id = None

permissions

save(**args*, ***kwargs*)

type = None

username = None

sevenbridges.models.project module

class sevenbridges.models.project.**Project**(***kwargs*)

Bases: *Resource*

Central resource for managing projects.

add_files(*files*)

Adds files to this project. :param files: List of files or a Collection object.

add_member(*user*, *permissions*)

Add a member to the project. :param user: Member username :param permissions: Permissions dictionary. :return: Member object.

add_member_division(*division*, *permissions*)

Add a member (team) to a project. :param division: Division object or division identifier. :param permissions: Permissions dictionary. :return: Member object.

add_member_email(*email*, *permissions=None*)

Add a member to the project using member email. :param email: Member email. :param permissions: Permissions dictionary. :return: Member object.

add_member_team(*team*, *permissions*)

Add a member (team) to a project. :param team: Team object or team identifier. :param permissions: Permissions dictionary. :return: Member object.

billing_group = None

category = None

classmethod create(*name*, *billing_group=None*, *description=None*, *tags=None*, *settings=None*, *api=None*)

Create a project. :param name: Project name. :param billing_group: Project billing group. :param description: Project description. :param tags: Project tags. :param settings: Project settings. :param api: Api instance. :return:

create_task(*name*, *app*, *revision=None*, *batch_input=None*, *batch_by=None*, *inputs=None*, *description=None*, *run=False*, *disable_batch=False*, *interruptible=True*, *execution_settings=None*)

Creates a task for this project.

Parameters

- **name** – Task name.
- **app** – CWL app identifier.
- **revision** – CWL app revision.
- **batch_input** – Batch input.
- **batch_by** – Batch criteria.
- **inputs** – Input map.
- **description** – Task description.
- **run** – True if you want to run a task upon creation.
- **disable_batch** – True if you want to disable batching.
- **interruptible** – True if you want to use interruptible instances.
- **execution_settings** – Execution settings for the task.

Returns

Task object.

created_by = None

created_on = None

deepcopy()

description = None

equals(*other*)

get_apps(*offset=None*, *limit=None*)

Retrieves apps in this project. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_exports(*volume=None*, *state=None*, *offset=None*, *limit=None*)

Fetches exports for this volume. :param volume: Optional volume identifier. :param state: Optional state. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_files(*offset=None*, *limit=None*)

Retrieves files in this project. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_imports(*volume=None*, *state=None*, *offset=None*, *limit=None*)

Fetches imports for this project. :param volume: Optional volume identifier. :param state: Optional state. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_member(*username*, *api=None*)

Fetches information about a single project member :param username: Member name :param api: Api instance :return: Member object

get_members(*offset=None*, *limit=None*)

Retrieves project members. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_tasks(*status=None*, *offset=None*, *limit=None*)

Retrieves tasks in this project. :param status: Optional task status. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

href = None

id = None

modified_on = None

name = None

classmethod query(*owner=None*, *name=None*, *offset=None*, *limit=None*, *api=None*, *category=None*, *tags=None*)

Query (List) projects :param owner: Owner username. :param name: Project name :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :param category: Project category. :param tags: Project tags. :return: Collection object.

remove_member(*user*)

Remove member from the project. :param user: User to be removed.

root_folder = None

save(**args*, ***kwargs*)

settings

tags = None

type = None

sevenbridges.models.rate_limit module

class sevenbridges.models.rate_limit.**RateLimit**(***kwargs*)

Bases: *Resource*

Rate limit resource contains info regarding request and computation rate limits.

deepcopy()

equals(*other*)

classmethod get(*id=None*, *api=None*)

Fetches the resource from the server. :param id: Resource identifier :param api: sevenbridges Api instance. :return: Resource object.

instance_limit

rate

sevenbridges.models.storage_export module

class sevenbridges.models.storage_export.**Export**(***kwargs*)

Bases: *Resource*

Central resource for managing exports.

classmethod bulk_get(*exports*, *api=None*)

Retrieve exports in bulk. :param exports: Exports to be retrieved. :param api: Api instance. :return: list of ExportBulkRecord objects.

classmethod bulk_submit(*exports*, *copy_only=False*, *api=None*)

Create exports in bulk. :param exports: List of dicts describing a wanted export. :param copy_only: If true files are kept on SevenBridges bucket. :param api: Api instance. :return: list of ExportBulkRecord objects.

deepcopy()

destination

equals(*other*)

error

finished_on = None

href = None

id = None

overwrite = None

properties

classmethod query(*volume=None*, *state=None*, *offset=None*, *limit=None*, *api=None*)

Query (List) exports. :param volume: Optional volume identifier. :param state: Optional import sate. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

property result

property source

started_on = None

state = None

classmethod submit_export(*file*, *volume*, *location*, *properties=None*, *overwrite=False*, *copy_only=False*, *api=None*)

Submit new export job. :param file: File to be exported. :param volume: Volume identifier. :param location: Volume location. :param properties: Properties dictionary. :param overwrite: If true it will overwrite file if exists :param copy_only: If true files are kept on SevenBridges bucket. :param api: Api Instance. :return: Export object.

class sevenbridges.models.storage_export.ExportBulkRecord(**kwargs)

Bases: BulkRecord

deepcopy()

equals(*other*)

resource

sevenbridges.models.storage_import module

class sevenbridges.models.storage_import.Import(**kwargs)

Bases: Resource

Central resource for managing imports.

autorename = None

classmethod bulk_get(*imports*, *api=None*)

Retrieve imports in bulk :param imports: Imports to be retrieved. :param api: Api instance. :return: List of ImportBulkRecord objects.

classmethod bulk_submit(*imports*, *api=None*)

Submit imports in bulk :param imports: List of dicts describing a wanted import. :param api: Api instance. :return: List of ImportBulkRecord objects.

deepcopy()

destination

equals(*other*)

error

finished_on = None

href = None

id = None

overwrite = None

preserve_folder_structure = None

classmethod query(*project=None*, *volume=None*, *state=None*, *offset=None*, *limit=None*, *api=None*)

Query (List) imports. :param project: Optional project identifier. :param volume: Optional volume identifier. :param state: Optional import sate. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

property result

source

started_on = None

state = None

classmethod submit_import(*volume*, *location*, *project=None*, *name=None*, *overwrite=False*, *properties=None*, *parent=None*, *preserve_folder_structure=True*, *autorename=False*, *api=None*)

Submits new import job. :param volume: Volume identifier. :param location: Volume location. :param project: Project identifier. :param name: Optional file name. :param overwrite: If true it will overwrite file if exists. :param properties: Properties dictionary. :param parent: The ID of the target folder to which the item should be imported. Should not be used together with project. :param preserve_folder_structure: Whether to keep the exact source folder structure. The default value is true if the item being imported is a folder. Should not be used if you are importing a file. :param autorename: Whether to automatically rename the item (by prefixing its name with an underscore and number) if another one with the same name already exists at the destination. :param api: Api instance. :return: Import object.

class sevenbridges.models.storage_import.**ImportBulkRecord**(***kwargs*)

Bases: *BulkRecord*

deepcopy()

equals(*other*)

resource

sevenbridges.models.task module

class sevenbridges.models.task.**Task**(***kwargs*)

Bases: *Resource*

Central resource for managing tasks.

abort(**args*, ***kwargs*)

app = None

batch = None

batch_by

batch_group

batch_input = None

classmethod bulk_get(*tasks*, *api=None*)

Retrieve tasks with specified ids in bulk :param tasks: Tasks to be retrieved. :param api: Api instance. :return: List of TaskBulkRecord objects.

clone(*run=True*)

Clone task :param run: run task after cloning :return: Task object.

classmethod create(*name*, *project*, *app*, *revision=None*, *batch_input=None*, *batch_by=None*, *inputs=None*, *description=None*, *run=False*, *disable_batch=False*, *interruptible=None*, *execution_settings=None*, *output_location=None*, *api=None*)

Creates a task on server. :param name: Task name. :param project: Project identifier. :param app: CWL app identifier. :param revision: CWL app revision. :param batch_input: Batch input. :param batch_by: Batch criteria. :param inputs: Input map. :param description: Task description. :param run: True if you want to run a task upon creation. :param disable_batch: If True disables batching of a batch task. :param interruptible: If True interruptible instance will be used. :param execution_settings: Execution settings for the task. :param output_location: Dictionary that allows you to define the exact location where your task outputs will be stored. :param api: Api instance. :return: Task object. :raises: TaskValidationError if validation Fails. :raises: SbgError if any exception occurs during request.

created_by = None

created_time = None

deepcopy()

description = None

end_time = None

equals(*other*)

errors = None

executed_by = None

execution_settings = None

execution_status

get_batch_children(*status=None*, *created_from=None*, *created_to=None*, *started_from=None*, *started_to=None*, *ended_from=None*, *ended_to=None*, *order_by=None*, *order=None*, *offset=None*, *limit=None*, *api=None*)

Retrieves batch child tasks for this task if its a batch task. :return: Collection instance. :raises SbError if task is not a batch task.

get_execution_details()

Retrieves execution details for a task. :return: Execution details instance.

href = None

id = None

inputs

name = None

origin = None

output_location = None

outputs

parent = None

price

project = None

classmethod query(*project=None*, *status=None*, *batch=None*, *parent=None*, *created_from=None*, *created_to=None*, *started_from=None*, *started_to=None*, *ended_from=None*, *ended_to=None*, *offset=None*, *limit=None*, *order_by=None*, *order=None*, *origin=None*, *api=None*)

Query (List) tasks. Date parameters may be both strings and python date objects. :param project: Target project. optional. :param status: Task status. :param batch: Only batch tasks. :param parent: Parent batch task identifier. :param ended_to: All tasks that ended until this date. :param ended_from: All tasks that ended from this date. :param started_to: All tasks that were started until this date. :param started_from: All tasks that were started from this date. :param created_to: All tasks that were created until this date. :param created_from: All tasks that were created from this date. :param offset: Pagination offset. :param limit: Pagination limit. :param order_by: Property to order by. :param order: Ascending or descending ordering. :param origin: Entity that created the task, e.g. automation run, if task was created by an automation run. :param api: Api instance. :return: Collection object.

run(**args*, ***kwargs*)

save(**args*, ***kwargs*)

start_time = None

status = None

type = None

use_interruptible_instances = None

wait(*period=10*, *callback=None*, **args*, ***kwargs*)

Wait until task is complete :param period: Time in seconds between reloads :param callback: Function to call after the task has finished, arguments and keyword arguments can be provided for it :return: Return value of provided callback function or None if a callback function was not provided

warnings = None

class sevenbridges.models.task.**TaskBulkRecord**(***kwargs*)

Bases: *BulkRecord*

deepcopy()

equals(*other*)

resource

sevenbridges.models.team module

class sevenbridges.models.team.**Team**(***kwargs*)

Bases: *Resource*

Central resource for managing teams.

add_member(*user*)

Add member to team :param user: User object or user's username :return: Added user.

classmethod create(*name*, *division*, *api=None*)

Create team within a division :param name: Team name. :param division: Parent division. :param api: Api instance. :return: Team object.

deepcopy()

equals(*other*)

get_members(*offset=None*, *limit=None*)

Fetch team members for current team. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

href = None

id = None

name = None

classmethod query(*division*, *list_all=False*, *offset=None*, *limit=None*, *api=None*)

Parameters

- **division** – Division slug.
- **list_all** – List all teams in division.
- **offset** – Pagination offset.
- **limit** – Pagination limit.
- **api** – Api instance.

Returns
Collection object.

remove_member(*user*)

Remove member from the team. :param user: User to be removed.

save(**args*, ***kwargs*)

sevenbridges.models.team_member module

class sevenbridges.models.team_member.**TeamMember**(***kwargs*)

Bases: *Resource*

Central resource for managing team members.

deepcopy()

equals(*other*)

href = None

id = None

role = None

username = None

sevenbridges.models.user module

class sevenbridges.models.user.**User**(***kwargs*)

Bases: *Resource*

Central resource for managing users.

address = None

affiliation = None

city = None

country = None

deepcopy()

disable(*api=None*)

Disable user :param api: Api instance. :return:

email = None

equals(*other*)

first_name = None

classmethod get(*user*, *api=None*)

Fetches the resource from the server. :param id: Resource identifier :param api: sevenbridges Api instance. :return: Resource object.

href = None

last_name = None

classmethod me(*api=None*)

Retrieves current user information. :param api: Api instance. :return: User object.

phone = None

classmethod query(*division*, *role=None*, *offset=None*, *limit=None*, *api=None*)

Query division users :param division: Division slug. :param role: User role in division. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

role = None

state = None

username = None

zip_code = None

sevenbridges.models.volume module

class sevenbridges.models.volume.**Volume**(***kwargs*)

Bases: *Resource*

Central resource for managing volumes.

access_mode = None

active = None

add_member(*user*, *permissions*)

Add a member to the volume. :param user: Member username :param permissions: Permissions dictionary. :return: Member object.

add_member_division(*division*, *permissions*)

Add a member (team) to a volume. :param division: Division object or division identifier. :param permissions: Permissions dictionary. :return: Member object.

add_member_team(*team*, *permissions*)

Add a member (team) to a volume. :param team: Team object or team identifier. :param permissions: Permissions dictionary. :return: Member object.

classmethod create_google_iam_volume(*name*, *bucket*, *configuration*, *access_mode*, *description=None*, *prefix=None*, *properties=None*, *api=None*)

Create google volume. :param name: Volume name. :param bucket: Referenced bucket. :param configuration: Google configuration. :param access_mode: Access Mode. :param description: Volume description. :param prefix: Volume prefix. :param properties: Volume properties. :param api: Api instance. :return: Volume object.

classmethod create_google_volume(*name*, *bucket*, *client_email*, *private_key*, *access_mode*, *description=None*, *prefix=None*, *properties=None*, *api=None*)

Create google volume. :param name: Volume name. :param bucket: Referenced bucket. :param client_email: Google client email. :param private_key: Google client private key. :param access_mode: Access Mode. :param description: Volume description. :param prefix: Volume prefix. :param properties: Volume properties. :param api: Api instance. :return: Volume object.

classmethod create_oss_volume(*name*, *bucket*, *endpoint*, *access_key_id*, *secret_access_key*, *access_mode*, *description=None*, *prefix=None*, *properties=None*, *api=None*)

Create oss volume. :param name: Volume name. :param bucket: Referenced bucket. :param access_key_id: Access key identifier. :param secret_access_key: Secret access key. :param access_mode: Access Mode. :param endpoint: Volume Endpoint. :param description: Volume description. :param prefix: Volume prefix. :param properties: Volume properties. :param api: Api instance. :return: Volume object.

classmethod create_s3_volume(*name*, *bucket*, *access_key_id*, *secret_access_key*, *access_mode*, *description=None*, *prefix=None*, *properties=None*, *api=None*)

Create s3 volume. :param name: Volume name. :param bucket: Referenced bucket. :param access_key_id: Amazon access key identifier. :param secret_access_key: Amazon secret access key. :param access_mode: Access Mode. :param description: Volume description. :param prefix: Volume prefix. :param properties: Volume properties. :param api: Api instance. :return: Volume object.

classmethod create_s3_volume_role_auth(*name*, *bucket*, *role_arn*, *external_id*, *access_mode*, *description=None*, *prefix=None*, *properties=None*, *api=None*)

Create s3 volume using IAM Role auth. :param name: Volume name. :param bucket: Referenced bucket. :param role_arn: Amazon role ARN. :param external_id: Amazon role external id. :param access_mode: Access Mode. :param description: Volume description. :param prefix: Volume prefix. :param properties: Volume properties. :param api: Api instance. :return: Volume object.

created_on = None

deepcopy()

description = None

equals(*other*)

get_exports(*state=None*, *offset=None*, *limit=None*)

Fetches exports for this volume. :param state: Optional state. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_imports(*project=None*, *state=None*, *offset=None*, *limit=None*)

Fetches imports for this volume. :param project: Optional project identifier. :param state: Optional state. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_member(*username*, *api=None*)

Fetches information about a single volume member :param username: Member name :param api: Api instance :return: Member object

get_members(*offset=None*, *limit=None*)

Retrieves volume members. :param offset: Pagination offset. :param limit: Pagination limit. :return: Collection object.

get_volume_object_info(*location*)

Fetches information about single volume object - usually file :param location: object location :return:

href = None

id = None

list(*prefix=None*, *limit=None*, *fields='_all'*)

modified_on = None

name = None

classmethod query(*offset=None*, *limit=None*, *api=None*)

Query (List) volumes. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

remove_member(*user*)

Remove member from the volume. :param user: User to be removed.

save(**args*, ***kwargs*)

service

sevenbridges.transfer package

Submodules

sevenbridges.transfer.download module

class sevenbridges.transfer.download.**DPartedFile**(*file_path*, *session*, *url*, *file_size*, *part_size*, *timeout*, *pool*)

Bases: object

done()

get_parts()

Partitions the file and saves the part information in memory.

submit()

Partitions the file into chunks and submits them into group of 4 for download on the api download pool.

class sevenbridges.transfer.download.**Download**(*url*, *file_path*, *part_size=None*, *retry_count=None*, *timeout=None*, *api=None*)

Bases: Thread

add_callback(*callback=None*, *errorback=None*)

Adds a callback that will be called when the download finishes successfully or when error is raised.

add_progress_callback(*callback=None*)

Adds a progress callback that will be called each time a get_parts is successfully downloaded. The first argument of the progress callback will be a progress object described in sevenbridges.transfer.utils

Parameters
callback – Callback function

property duration

property path

pause()

Pauses the download. :raises SbgError: If upload is not in RUNNING state.

property progress

resume()

Resumes the download. :raises SbgError: If download is not in RUNNING state.

run()

Runs the thread! Should not be used use start() method instead.

start()

Starts the download. :raises SbgError: If download is not in PREPARING state.

property start_time

property status

stop()

Stops the download. :raises SbgError: If download is not in PAUSED or RUNNING state.

wait()

Blocks until download is completed.

sevenbridges.transfer.upload module

class sevenbridges.transfer.upload.**CodePackageUPartedFile**(*fp*, *file_size*, *part_size*, *upload*, *timeout*, *storage_session*, *api*)

Bases: *UPartedFile*

class sevenbridges.transfer.upload.**CodePackageUpload**(*file_path*, *automation_id*, *file_name=None*, *part_size=None*, *retry_count=6*, *timeout=300*, *parent=None*, *api=None*)

Bases: *Upload*

partition_file(*fp*)

class sevenbridges.transfer.upload.**UPartedFile**(*fp*, *file_size*, *part_size*, *upload*, *timeout*, *storage_session*, *api*)

Bases: object

done()

get_parts()

Partitions the file and saves the parts to be uploaded in memory.

submit()

Partitions the file into chunks and submits them into group of 4 for upload on the api upload pool. :return: Futures

class sevenbridges.transfer.upload.**Upload**(*file_path*, *project=None*, *parent=None*, *file_name=None*, *overwrite=False*, *part_size=None*, *retry_count=None*, *timeout=None*, *api=None*)

Bases: Thread

add_callback(*callback=None*, *errorback=None*)

Adds a callback that will be called when the upload finishes successfully or when error is raised.

add_progress_callback(*callback=None*)

Adds a progress callback that will be called each time a get_parts is successfully uploaded. The first argument of the progress callback will be a progress object described in sevenbridges.transfer.utils

Parameters
callback – Callback function

property duration

property file_name

partition_file(*fp*)

pause()

Pauses the upload. :raises SbgError: If upload is not in RUNNING state.

property progress

result()

resume()

Resumes the upload that was paused. :raises SbgError: If upload is not in PAUSED state.

run()

Runs the thread! Should not be used use start() method instead.

start()

Starts the upload. :raises SbgError: If upload is not in PREPARING state.

property start_time

property status

stop()

Stops the upload. :raises SbgError: If upload is not in PAUSED or RUNNING state.

wait()

Blocks until upload is completed.

sevenbridges.transfer.utils module

class sevenbridges.transfer.utils.**Part**(*start=None*, *size=None*)

Bases: object

property size

property start

class sevenbridges.transfer.utils.**Progress**(*num_of_parts*, *parts_done*, *bytes_done*, *file_size*, *duration*)

Bases: object

property bandwidth

property bytes_done

property duration

property file_size

property num_of_parts

property parts_done

property progress

sevenbridges.transfer.utils.**simple_progress_bar**(*progress*)

sevenbridges.transfer.utils.**total_parts**(*file_size*, *part_size*)

1.4.3 Submodules

1.4.4 sevenbridges.api module

class sevenbridges.api.**Api**(*url=None*, *token=None*, *oauth_token=None*, *config=None*, *timeout=None*, *download_max_workers=16*, *upload_max_workers=16*, *proxies=None*, *error_handlers=None*, *advance_access=False*, *pool_connections=10*, *pool_maxsize=100*, *pool_block=True*, *max_parallel_requests=100*, *retry_count=6*, *backoff_factor=1*, *debug=False*)

Bases: *HttpClient*

Api aggregates all resource classes into single place

actions
alias of *Actions*

apps
alias of *App*

async_jobs
alias of *AsyncJob*

automation_packages
alias of *AutomationPackage*

automation_runs
alias of *AutomationRun*

automations
alias of *Automation*

billing_groups
alias of *BillingGroup*

datasets
alias of *Dataset*

divisions
alias of *Division*

drs_imports
alias of *DRSImportBulk*

endpoints
alias of *Endpoints*

exports
alias of *Export*

files
alias of *File*

imports
alias of *Import*

invoices
alias of *Invoice*

markers
alias of *Marker*

projects
alias of *Project*

rate_limit
alias of *RateLimit*

tasks
alias of *Task*

teams
alias of *Team*

users
alias of *User*

volumes
alias of *Volume*

1.4.5 sevenbridges.config module

class sevenbridges.config.**Config**(*profile=None*, *proxies=None*, *advance_access=None*)
Bases: object

Utility configuration class.

class sevenbridges.config.**UserProfile**(*profile*)
Bases: object

CONFIG = '/home/docs/.sevenbridges/sevenbridges-python/config'

CREDENTIALS = '/home/docs/.sevenbridges/credentials'

property advance_access

property api_endpoint

property auth_token

property proxies

sevenbridges.config.**format_proxies**(*proxies*)
Helper method for request proxy key compatibility. :param proxies: Proxies dictionary :return: Dict compatible with request proxy format.

1.4.6 sevenbridges.decorators module

sevenbridges.decorators.**check_for_error**(*func*)
Executes the wrapped function and inspects the response object for specific errors.

sevenbridges.decorators.**inplace_reload**(*method*)
Executes the wrapped function and reloads the object with data returned from the server.

sevenbridges.decorators.**throttle**(*func*)
Throttles number of parallel requests made by threads from single HttpClient session.

1.4.7 sevenbridges.errors module

exception sevenbridges.errors.**AdvanceAccessError**(*message=None*)

Bases: *SbgError*

exception sevenbridges.errors.**BadRequest**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**Conflict**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**ExecutionDetailsInvalidTaskType**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**Forbidden**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**LocalFileAlreadyExists**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**MethodNotAllowed**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**NonJSONResponseError**(*status*, *code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**NotFound**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**PaginationError**(*message*)

Bases: *SbgError*

exception sevenbridges.errors.**ReadOnlyPropertyError**(*message*)

Bases: *SbgError*

exception sevenbridges.errors.**RequestTimeout**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**ResourceNotModified**

Bases: *SbgError*

exception sevenbridges.errors.**SbgError**(*message=None*, *code=None*, *status=None*, *more_info=None*)

Bases: `Exception`

Base class for SBG errors.

Provides a base exception for all errors that are thrown by sevenbridges-python library.

exception sevenbridges.errors.**ServerError**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**ServiceUnavailable**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**TaskValidationError**(*message*, *task=None*)

Bases: *SbgError*

exception sevenbridges.errors.**TooManyRequests**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**URITooLong**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**Unauthorized**(*code=None*, *message=None*, *more_info=None*)

Bases: *SbgError*

exception sevenbridges.errors.**ValidationError**(*message*)

Bases: *SbgError*

PYTHON MODULE INDEX

S

INDEX

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Z