sbg Documentation

Release 0.12.0

Seven Bridges Genomics

May 18, 2018

Contents

1 Content **3**
1.1 Installation 3
1.2 Get the Code 3
1.3 Quickstart 3
1.4 sevenbridges package 26

Python Module Index **79**

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 1

Content

Installation

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

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

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.

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.

Authentication and Configuration

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

1. Your authentication token
2. 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

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.

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]
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
```

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.

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
```

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.

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
```

For example, to obtain your email address invoke:

```
me.email
```

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)
```

```
# 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

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.

```python
# 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()`

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 available 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
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()
```

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

There is one billing group method:

`breakdown()` fetches a cost breakdown by project and analysis 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)

Managing files

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.

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.

`name` - File name.

`size` - File size in bytes.

`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()`

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'} )


# 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()

# 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')
```

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

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 - `create_s3_volume()`
- Create a new volume based on Google Cloud Storage service - `create_google_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
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
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'
)

# 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.

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)
      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
```

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. The sevenbridges-python currently supports only Draft 2 format of the CWL. Each app has a CWL description, expressed in JSON.

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.

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

App properties

Each app has the following available 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.

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_status` - Task execution status.

`price` - Task cost.

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

`outputs` - Generated outputs from the task.

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()`.

Task creation hints

- Both input files and parameters are passed the same way together in a single dictionary to `inputs`.
- `api.files.query` always return an array of files. For single file inputs, use `api.files.query(project='my-project', names=["one_file.fa"])[0]`.

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()
```

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.')
```

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)
```

Files

The following operations are supported:

- `bulk_get()` - Retrieves multiple files.
- `bulk_edit()` - Edits multiple files.
- `bulk_update()` - Updates multiple files.
- `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`.

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')

# Submit export jobs
exports = [
    {
        'file': 'example_file.txt',
        'volume': volume,
        'location': '/data/example_file.txt',
        'properties': {
            'some_property': 'value'
        }
        'overwrite': True
    },
    {
        'file': 'example_file_2.txt',
        'volume': volume,
        'location': '/data/example_file_2.txt',
        'properties': {
            'some_property_2': 'value_2'
        },
        'overwrite': False
    },
]
response = api.exports.bulk_submit(exports=exports)
```

sevenbridges package

sevenbridges-python

copyright 2018 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*)

Bases: *`sevenbridges.http.client.HttpClient`*

Api aggregates all resource classes into single place

`actions`
alias of `Actions`

`apps`
alias of *`App`*

`billing_groups`
alias of *`BillingGroup`*

`datasets`
alias of *`Dataset`*

`divisions`
alias of *`Division`*

`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.**Config**(*profile=None*, *proxies=None*, *advance_access=None*)
Bases: object

Utility configuration class.

class sevenbridges.**Invoice**(***kwargs*)
Bases: *sevenbridges.meta.resource.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

class sevenbridges.**BillingGroup**(***kwargs*)
Bases: *sevenbridges.meta.resource.Resource*

Central resource for managing billing groups.

balance

breakdown()
Get Billing group breakdown for the current billing group.

deepcopy()

disabled = None

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.

type = None

class sevenbridges.**BillingGroupBreakdown**(***kwargs*)
Bases: *sevenbridges.meta.resource.Resource*

Central resource for managing billing group breakdowns.

deepcopy()

equals(*other*)

href = None

project_breakdown

total_spending

class sevenbridges.**User**(***kwargs*)
Bases: *sevenbridges.meta.resource.Resource*

Central resource for managing tasks.

address = None

affiliation = None

city = None

country = None

deepcopy()

email = None

equals(*other*)

first_name = None

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

href = None

last_name = None

classmethod **me**(*api=None*)
Retrieves current user information. :param api: Api instance. :return: User object.

phone = None

state = None

username = None

zip_code = None

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

Bases: `sevenbridges.meta.resource.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**

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

Bases: `sevenbridges.meta.resource.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**

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*)
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.

Returns Task object.

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_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

name = None

classmethod query (*owner=None*, *offset=None*, *limit=None*, *api=None*)
Query (List) projects :param owner: Owner username. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

remove_member (*user*)
Remove member from the project. :param user: User to be removed.

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

settings

tags = None

type = None

class sevenbridges.**Task** (***kwargs*)
Bases: *sevenbridges.meta.resource.Resource*

Central resource for managing tasks.

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

app = None

batch = None

batch_by

batch_group

batch_input = None

classmethod create (*name*, *project*, *app*, *revision=None*, *batch_input=None*, *batch_by=None*, *inputs=None*, *description=None*, *run=False*, *disable_batch=False*, *interruptible=True*, *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 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_status

get_batch_children ()
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**

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*, *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 api: Api instance. :return: Collection object.

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

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

start_time = **None**

status = **None**

type = **None**

use_interruptible_instances = **None**

warnings = **None**

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

Bases: *sevenbridges.meta.resource.Resource*

Central resource for managing apps.

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

Copies the current app. :param project: Destination project. :param name: Destination app name. :return: Copied App object.

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**

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.**Member** (***kwargs*)

Bases: *sevenbridges.meta.resource.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 (*obj*, **args*, ***kwargs*)

type = None

username = None

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

Bases: *sevenbridges.meta.comp_mutable_dict.CompoundMutableDict*, *sevenbridges.meta.resource.Resource*

Members permissions resource.

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

Bases: *sevenbridges.meta.resource.Resource*

Central resource for managing files.

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*)

Edit files with specified ids in bulk :param files: Files to be updated. :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*)

Update files with specified ids in bulk :param files: Files to be updated. :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.

Parameters

- **`path`** – Path for file download If omitted tmp file will be used.
- **`overwrite`** – Overwrite file if exists locally
- **`encoding`** – File encoding, by default it is UTF-8

Returns File content.

`copy` (*project*, *name=None*)
Copies the current file. :param project: Destination project. :param name: Destination file name. :return: Copied File object.

`created_on` **= None**

`deepcopy` ()

`download` (*path*, *retry=5*, *timeout=10*, *chunk_size=5242880*, *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**

`metadata`

`modified_on` **= None**

`name` **= None**

`origin`

`project` **= None**

classmethod `query` (*project=None*, *names=None*, *metadata=None*, *origin=None*, *tags=None*, *offset=None*, *limit=None*, *dataset=None*, *api=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. :return: Collection object.

`reload` ()
Refreshes the file with the data from the server.

`save` (*obj*, **args*, ***kwargs*)

`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

classmethod **upload**(*path*, *project*, *file_name=None*, *overwrite=False*, *retry=5*, *timeout=10*, *part_size=5242880*, *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
- **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.**Export**(***kwargs*)
Bases: *sevenbridges.meta.resource.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*, *api=None*)
Create exports in bulk. :param exports: Exports to be submitted in bulk. :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 api: Api instance. :return: Collection object.

result

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.**Import**(***kwargs*)
Bases: sevenbridges.meta.resource.Resource

Central resource for managing imports.

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: Imports to be retrieved. :param api: Api instance. :return: List of ImportBulkRecord objects.

deepcopy()

destination

equals(*other*)

error

finished_on = None

href = None

id = None

overwrite = 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 api: Api instance. :return: Collection object.

result

source

started_on = None

state = None

classmethod **submit_import**(*volume*, *location*, *project*, *name=None*, *overwrite=False*, *properties=None*, *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 api: Api instance. :return: Import object.

class sevenbridges.**Volume**(***kwargs*)
Bases: sevenbridges.meta.resource.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_volume`** (*name*, *bucket*, *client_email*, *private_key*, *access_mode*, *description=None*, *prefix=None*, *properties=None*, *api=None*)
Create s3 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_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.

`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_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=50*)

`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`(*obj*, **args*, ***kwargs*)

`service`

class `sevenbridges.`**`Marker`**(***kwargs*)
Bases: `sevenbridges.meta.resource.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`()

`delete`()

`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.

`reload`()

`save`(*obj*, **args*, ***kwargs*)

class `sevenbridges.`**`Division`**(***kwargs*)
Bases: `sevenbridges.meta.resource.Resource`

Central resource for managing divisions.

`deepcopy`()

`equals`(*other*)

classmethod **`get`**(*id*, *api=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.

`reload`()

class `sevenbridges.`**`Dataset`**(***kwargs*)
Bases: *`sevenbridges.meta.resource.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`(*obj*, **args*, ***kwargs*)

class `sevenbridges.`**`BulkRecord`**(***kwargs*)
Bases: *`sevenbridges.meta.resource.Resource`*

`deepcopy`()

`equals`(*other*)

`error`

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

resource

valid

class sevenbridges.**TransferState**
Bases: object

ABORTED = 'ABORTED'

COMPLETED = 'COMPLETED'

FAILED = 'FAILED'

PAUSED = 'PAUSED'

PREPARING = 'PREPARING'

RUNNING = 'RUNNING'

STOPPED = 'STOPPED'

class sevenbridges.**VolumeType**
Bases: object

GOOGLE = 'GCS'

S3 = 'S3'

class sevenbridges.**VolumeAccessMode**
Bases: object

READ_ONLY = 'RO'

READ_WRITE = 'RW'

class sevenbridges.**FileStorageType**
Bases: object

PLATFORM = 'PLATFORM'

VOLUME = 'VOLUME'

class sevenbridges.**ImportExportState**
Bases: object

COMPLETED = 'COMPLETED'

FAILED = 'FAILED'

PENDING = 'PENDING'

RUNNING = 'RUNNING'

class sevenbridges.**TaskStatus**
Bases: object

ABORTED = 'ABORTED'

COMPLETED = 'COMPLETED'

CREATING = 'CREATING'

DRAFT = 'DRAFT'

FAILED = 'FAILED'

QUEUED = 'QUEUED'

RUNNING = 'RUNNING'

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.`**`ResourceNotModified`**
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`ReadOnlyPropertyError`**(*message*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`ValidationError`**(*message*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`TaskValidationError`**(*message*, *task=None*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`PaginationError`**(*message*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`BadRequest`**(*code=None*, *message=None*, *more_info=None*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`Unauthorized`**(*code=None*, *message=None*, *more_info=None*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`Forbidden`**(*code=None*, *message=None*, *more_info=None*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`NotFound`**(*code=None*, *message=None*, *more_info=None*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`Conflict`**(*code=None*, *message=None*, *more_info=None*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`TooManyRequests`**(*code=None*, *message=None*, *more_info=None*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`ServerError`**(*code=None*, *message=None*, *more_info=None*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`ServiceUnavailable`**(*code=None*, *message=None*, *more_info=None*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`MethodNotAllowed`**(*code=None*, *message=None*, *more_info=None*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`RequestTimeout`**(*code=None*, *message=None*, *more_info=None*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`LocalFileAlreadyExists`**(*code=None*, *message=None*, *more_info=None*)
Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.`**`ExecutionDetailsInvalidTaskType`**(*code=None*, *message=None*, *more_info=None*)
Bases: *`sevenbridges.errors.SbgError`*

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*)

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*)

`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*)

`remaining`

`remove_error_handler`(*handler*)

`request_id`

`reset_time`

`session`

`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`**(*proxies=None*)

Utility method to generate request sessions. :param proxies: Proxies dictionary. :return: requests.Session object.

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.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**

`total`

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

Bases: *`sevenbridges.meta.collection.Collection`*

`next_page`()

Fetches next result set. :return: VolumeCollection object.

`previous_page`()

`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`()

`update`(*E=None*, ***F*)

sevenbridges.meta.data module

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

Bases: `object`

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

fetch()

sevenbridges.meta.fields module

class sevenbridges.meta.fields.**BasicListField**(*name=None*, *read_only=False*, *max_length=None*)
Bases: *sevenbridges.meta.fields.Field*

validate(*value*)

class sevenbridges.meta.fields.**BooleanField**(*name=None*, *read_only=False*)
Bases: *sevenbridges.meta.fields.Field*

validate(*value*)

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

class sevenbridges.meta.fields.**CompoundListField**(*cls*, *name=None*, *read_only=True*)
Bases: *sevenbridges.meta.fields.Field*

class sevenbridges.meta.fields.**DateTimeField**(*name=None*, *read_only=True*)
Bases: *sevenbridges.meta.fields.Field*

class sevenbridges.meta.fields.**DictField**(*name=None*, *read_only=False*)
Bases: *sevenbridges.meta.fields.Field*, dict

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

validate(*value*)

class sevenbridges.meta.fields.**FloatField**(*name=None*, *read_only=False*)
Bases: *sevenbridges.meta.fields.Field*

validate(*value*)

class sevenbridges.meta.fields.**HrefField**(*name=None*)
Bases: *sevenbridges.meta.fields.Field*

class sevenbridges.meta.fields.**IntegerField**(*name=None*, *read_only=False*)
Bases: *sevenbridges.meta.fields.Field*

validate(*value*)

class sevenbridges.meta.fields.**ObjectIdField**(*name=None*, *read_only=True*)
Bases: *sevenbridges.meta.fields.Field*

class sevenbridges.meta.fields.**StringField**(*name=None*, *read_only=False*, *max_length=None*)
Bases: *sevenbridges.meta.fields.Field*

validate(*value*)

class sevenbridges.meta.fields.**UuidField**(*name=None*, *read_only=True*)
Bases: *sevenbridges.meta.fields.Field*

validate(*value*)

sevenbridges.meta.resource module

class `sevenbridges.meta.resource.`**`Resource`**(*api*)

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.

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.

class `sevenbridges.meta.resource.`**`ResourceMeta`**

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_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_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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.comp_mutable_dict.CompoundMutableDict`*, *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.comp_mutable_dict.CompoundMutableDict`*, *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.comp_mutable_dict.CompoundMutableDict`*, *`sevenbridges.meta.resource.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: *`sevenbridges.meta.comp_mutable_dict.CompoundMutableDict`*, *`sevenbridges.meta.resource.Resource`*

Members permissions resource.

sevenbridges.models.compound.projects.settings module

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

Bases: *`sevenbridges.meta.comp_mutable_dict.CompoundMutableDict`*, *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.Resource`*, `dict`

Task batch by resource.

`equals`(*other*)

`update`(*E=None*, ***F*)

sevenbridges.models.compound.tasks.batch_group module

class `sevenbridges.models.compound.tasks.batch_group.`**`BatchGroup`**(***kwargs*)

Bases: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *sevenbridges.meta.comp_mutable_dict.CompoundMutableDict*, *sevenbridges.meta.resource.Resource*

Task input resource.

sevenbridges.models.compound.tasks.output module

class sevenbridges.models.compound.tasks.output.**Output**(***kwargs*)

Bases: *sevenbridges.meta.comp_mutable_dict.CompoundMutableDict*, *sevenbridges.meta.resource.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: *sevenbridges.meta.resource.Resource*

ImportDestination resource describes the location of the file imported on to SevenBridges platform or related product.

deepcopy()

equals(*other*)

name = None

project = None

sevenbridges.models.compound.volumes.properties module

class `sevenbridges.models.compound.volumes.properties.`**`VolumeProperties`**`(`***kwargs*`)`

Bases: *`sevenbridges.meta.comp_mutable_dict.CompoundMutableDict`*, *`sevenbridges.meta.resource.Resource`*

Volume permissions resource.

sevenbridges.models.compound.volumes.service module

class `sevenbridges.models.compound.volumes.service.`**`VolumeService`**`(`***kwargs*`)`

Bases: *`sevenbridges.meta.comp_mutable_dict.CompoundMutableDict`*, *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.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: *sevenbridges.meta.resource.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: *sevenbridges.meta.resource.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: *sevenbridges.meta.resource.Resource*

Breakdown resource contains price breakdown by storage and computation.

computation = None

deepcopy()

equals(*other*)

storage = None

Submodules

sevenbridges.models.actions module

class `sevenbridges.models.actions.`**`Actions`**(***kwargs*)

Bases: `sevenbridges.meta.resource.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: `sevenbridges.meta.resource.Resource`

Central resource for managing apps.

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

Copies the current app. :param project: Destination project. :param name: Destination app name. :return: Copied App object.

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**

`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_breakdown module

class sevenbridges.models.billing_breakdown.**BillingGroupBreakdown**(***kwargs*)
Bases: *sevenbridges.meta.resource.Resource*

Central resource for managing billing group breakdowns.

deepcopy()

equals(*other*)

href = None

project_breakdown

total_spending

sevenbridges.models.billing_group module

class sevenbridges.models.billing_group.**BillingGroup**(***kwargs*)
Bases: *sevenbridges.meta.resource.Resource*

Central resource for managing billing groups.

balance

breakdown()
Get Billing group breakdown for the current billing group.

deepcopy()

disabled = None

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.

type = None

sevenbridges.models.division module

class `sevenbridges.models.division.`**`Division`**`(`***kwargs*`)`

Bases: *`sevenbridges.meta.resource.Resource`*

Central resource for managing divisions.

`deepcopy``()`

`equals``(`*other*`)`

classmethod `get``(`*id*, *api=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.

`reload``()`

sevenbridges.models.endpoints module

class `sevenbridges.models.endpoints.`**`Endpoints`**`(`***kwargs*`)`

Bases: *`sevenbridges.meta.resource.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.**AppRawFormat**
Bases: object

JSON = 'json'

YAML = 'yaml'

class sevenbridges.models.enums.**FeedbackType**
Bases: object

IDEA = 'IDEA'

PROBLEM = 'PROBLEM'

THOUGHT = 'THOUGHT'

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

class sevenbridges.models.enums.**TaskStatus**
Bases: object

ABORTED = 'ABORTED'

COMPLETED = 'COMPLETED'

CREATING = 'CREATING'

DRAFT = 'DRAFT'

FAILED = 'FAILED'

QUEUED = 'QUEUED'

RUNNING = 'RUNNING'

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'

S3 = 'S3'

sevenbridges.models.execution_details module

class sevenbridges.models.execution_details.**ExecutionDetails**(***kwargs*)
Bases: *sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.Resource`*

Central resource for managing files.

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*)

Edit files with specified ids in bulk :param files: Files to be updated. :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*)

Update files with specified ids in bulk :param files: Files to be updated. :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.

Parameters

- **`path`** – Path for file download If omitted tmp file will be used.
- **`overwrite`** – Overwrite file if exists locally
- **`encoding`** – File encoding, by default it is UTF-8

Returns File content.

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

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

`created_on` **= None**

`deepcopy` ()

`download` (*path*, *retry=5*, *timeout=10*, *chunk_size=5242880*, *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**

metadata

modified_on = None

name = None

origin

project = None

classmethod query(*project=None*, *names=None*, *metadata=None*, *origin=None*, *tags=None*, *offset=None*, *limit=None*, *dataset=None*, *api=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. :return: Collection object.

reload()

Refreshes the file with the data from the server.

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

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

classmethod upload(*path*, *project*, *file_name=None*, *overwrite=False*, *retry=5*, *timeout=10*, *part_size=5242880*, *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
- **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: sevenbridges.models.bulk.BulkRecord

deepcopy()

equals(*other*)

resource

sevenbridges.models.invoice module

class `sevenbridges.models.invoice.`**`Invoice`**(***kwargs*)

Bases: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.Resource`*

Pagination links.

`deepcopy`()

`equals`(*other*)

`href` = **None**

`method` = **None**

`rel` = **None**

class `sevenbridges.models.link.`**`VolumeLink`**(***kwargs*)

Bases: *`sevenbridges.meta.resource.Resource`*

Pagination links for volumes.

`deepcopy`()

`equals`(*other*)

`next` = **None**

sevenbridges.models.marker module

class `sevenbridges.models.marker.`**`Marker`**(***kwargs*)

Bases: *`sevenbridges.meta.resource.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`()

`delete`()

`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.

`reload`()

`save`(*obj*, **args*, ***kwargs*)

sevenbridges.models.member module

class `sevenbridges.models.member.`**`Member`**(***kwargs*)
Bases: *`sevenbridges.meta.resource.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`(*obj*, **args*, ***kwargs*)

`type` = None

`username` = None

sevenbridges.models.project module

class `sevenbridges.models.project.`**`Project`**(***kwargs*)
Bases: *`sevenbridges.meta.resource.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

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*)
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.

Returns Task object.

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_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**

`name` = **None**

classmethod `query`(*owner=None*, *offset=None*, *limit=None*, *api=None*)
Query (List) projects :param owner: Owner username. :param offset: Pagination offset. :param limit: Pagination limit. :param api: Api instance. :return: Collection object.

`remove_member`(*user*)
Remove member from the project. :param user: User to be removed.

`save`(*obj*, **args*, ***kwargs*)

`settings`

`tags` = **None**

`type` = **None**

sevenbridges.models.rate_limit module

class `sevenbridges.models.rate_limit.`**`RateLimit`**(***kwargs*)
Bases: `sevenbridges.meta.resource.Resource`

Rate limit resource contains info regarding request and computation rate limits.

`deepcopy`()

`equals`(*other*)

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

`instance_limit`

`rate`

sevenbridges.models.storage_export module

class `sevenbridges.models.storage_export.`**`Export`**(***kwargs*)

Bases: `sevenbridges.meta.resource.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*, *api=None*)

Create exports in bulk. :param exports: Exports to be submitted in bulk. :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 api: Api instance. :return: Collection object.

`result`

`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: `sevenbridges.models.bulk.BulkRecord`

`deepcopy`()

`equals`(*other*)

`resource`

sevenbridges.models.storage_import module

class `sevenbridges.models.storage_import.`**`Import`**(***kwargs*)

Bases: *`sevenbridges.meta.resource.Resource`*

Central resource for managing imports.

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: Imports to be retrieved. :param api: Api instance. :return: List of ImportBulkRecord objects.

`deepcopy`()

`destination`

`equals`(*other*)

`error`

`finished_on` = None

`href` = None

`id` = None

`overwrite` = 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 api: Api instance. :return: Collection object.

`result`

`source`

`started_on` = None

`state` = None

classmethod **`submit_import`**(*volume*, *location*, *project*, *name=None*, *overwrite=False*, *properties=None*, *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 api: Api instance. :return: Import object.

class `sevenbridges.models.storage_import.`**`ImportBulkRecord`**(***kwargs*)

Bases: `sevenbridges.models.bulk.BulkRecord`

`deepcopy`()

`equals`(*other*)

`resource`

sevenbridges.models.task module

class `sevenbridges.models.task.`**`Task`**(***kwargs*)

Bases: *`sevenbridges.meta.resource.Resource`*

Central resource for managing tasks.

`abort`(*obj*, **args*, ***kwargs*)

`app` = **None**

`batch` = **None**

`batch_by`

`batch_group`

`batch_input` = **None**

classmethod `create`(*name*, *project*, *app*, *revision=None*, *batch_input=None*, *batch_by=None*, *inputs=None*, *description=None*, *run=False*, *disable_batch=False*, *interruptible=True*, *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 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_status`

`get_batch_children`()

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**

`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*, *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 api: Api instance. :return: Collection object.

`run`(*obj*, **args*, ***kwargs*)

`save`(*obj*, **args*, ***kwargs*)

`start_time` = **None**

`status` = **None**

`type` = **None**

`use_interruptible_instances` = **None**

`warnings` = **None**

sevenbridges.models.team module

class `sevenbridges.models.team.`**`Team`**(***kwargs*)

Bases: `sevenbridges.meta.resource.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`()

`delete`()

`equals`(*other*)

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

`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*, *offset=None*, *limit=None*, *api=None*)

`reload`()

`remove_member`(*user*)

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

`save`(*obj*, **args*, ***kwargs*)

sevenbridges.models.team_member module

class `sevenbridges.models.team_member.`**`TeamMember`**`(**kwargs)`

Bases: *`sevenbridges.meta.resource.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: *`sevenbridges.meta.resource.Resource`*

Central resource for managing tasks.

`address` = **None**

`affiliation` = **None**

`city` = **None**

`country` = **None**

`deepcopy``()`

`email` = **None**

`equals``(other)`

`first_name` = **None**

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

`href` = **None**

`last_name` = **None**

classmethod `me``(api=None)`

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

`phone` = **None**

`state` = **None**

`username` = **None**

`zip_code` = **None**

sevenbridges.models.volume module

class `sevenbridges.models.volume.`**`Volume`**(***kwargs*)

Bases: `sevenbridges.meta.resource.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_volume`**(*name*, *bucket*, *client_email*, *private_key*, *access_mode*, *description=None*, *prefix=None*, *properties=None*, *api=None*)

Create s3 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_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.

`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_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=50*)

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 (*obj*, **args*, ***kwargs*)

service

sevenbridges.transfer package

Submodules

sevenbridges.transfer.download module

class sevenbridges.transfer.download.**DPartedFile** (*file_path*, *session*, *url*, *file_size*, *part_size*, *retry*, *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=5242880*, *retry_count=5*, *timeout=60*, *api=None*)
Bases: threading.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

duration

path

pause ()
Pauses the download. :raises SbgError: If upload is not in RUNNING state.

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.

start_time

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.**UPartedFile**(*fp*, *file_size*, *part_size*, *upload*, *retry_count*, *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*, *file_name=None*, *overwrite=False*, *part_size=5242880*, *retry_count=5*, *timeout=60*, *api=None*)
Bases: `threading.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

duration

file_name

pause()
Pauses the upload. :raises SbgError: If upload is not in RUNNING state.

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.

start_time

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

size

start

class sevenbridges.transfer.utils.**Progress**(*num_of_parts*, *parts_done*, *bytes_done*, *file_size*, *duration*)
Bases: object

bandwidth

bytes_done

duration

file_size

num_of_parts

parts_done

progress

sevenbridges.transfer.utils.**simple_progress_bar**(*progress*)

sevenbridges.transfer.utils.**total_parts**(*file_size*, *part_size*)

Submodules

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*)
Bases: *sevenbridges.http.client.HttpClient*

Api aggregates all resource classes into single place

actions
alias of Actions

apps
alias of App

billing_groups
alias of `BillingGroup`

datasets
alias of `Dataset`

divisions
alias of `Division`

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`

sevenbridges.config module

class `sevenbridges.config.`**`Config`**(*profile=None*, *proxies=None*, *advance_access=None*)
Bases: `object`

Utility configuration class.

class `sevenbridges.config.`**`Profile`**(*profile*)
Bases: `object`

CONFIG = '/home/docs/.sevenbridges/sevenbridges-python/config'

CREDENTIALS = '/home/docs/.sevenbridges/credentials'

advance_access

api_endpoint

auth_token

proxies

sevenbridges.config.**format_proxies**(*proxies*)

Helper method for request proxy key compatibility. :param proxies: Proxies dictionary :return: Dict compatible with request proxy format.

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.**retry**(*retry_count*)

Retry decorator used during file upload and download.

sevenbridges.decorators.**retry_on_excs**(*excs*, *retry_count=3*, *delay=5*)

Retry decorator used to retry callables on for specific exceptions.

Parameters

- **excs** – Exceptions tuple.
- **retry_count** – Retry count.
- **delay** – Delay in seconds between retries.

Returns Wrapped function object.

sevenbridges.errors module

exception sevenbridges.errors.**AdvanceAccessError**(*message=None*)

Bases: *sevenbridges.errors.SbgError*

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

Bases: *sevenbridges.errors.SbgError*

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

Bases: *sevenbridges.errors.SbgError*

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

Bases: *sevenbridges.errors.SbgError*

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

Bases: *sevenbridges.errors.SbgError*

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

Bases: *sevenbridges.errors.SbgError*

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

Bases: *sevenbridges.errors.SbgError*

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

Bases: *sevenbridges.errors.SbgError*

exception `sevenbridges.errors.`**`PaginationError`**(*message*)
  Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.errors.`**`ReadOnlyPropertyError`**(*message*)
  Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.errors.`**`RequestTimeout`**(*code=None*, *message=None*, *more_info=None*)
  Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.errors.`**`ResourceNotModified`**
  Bases: *`sevenbridges.errors.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: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.errors.`**`ServiceUnavailable`**(*code=None*, *message=None*, *more_info=None*)
  Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.errors.`**`TaskValidationError`**(*message*, *task=None*)
  Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.errors.`**`TooManyRequests`**(*code=None*, *message=None*, *more_info=None*)
  Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.errors.`**`Unauthorized`**(*code=None*, *message=None*, *more_info=None*)
  Bases: *`sevenbridges.errors.SbgError`*

exception `sevenbridges.errors.`**`ValidationError`**(*message*)
  Bases: *`sevenbridges.errors.SbgError`*

Python Module Index

S

sevenbridges, 26
sevenbridges.api, 74
sevenbridges.config, 75
sevenbridges.decorators, 76
sevenbridges.errors, 76
sevenbridges.http, 42
sevenbridges.http.client, 42
sevenbridges.http.error_handlers, 42
sevenbridges.meta, 43
sevenbridges.meta.collection, 43
sevenbridges.meta.comp_mutable_dict, 43
sevenbridges.meta.data, 43
sevenbridges.meta.fields, 44
sevenbridges.meta.resource, 45
sevenbridges.meta.transformer, 45
sevenbridges.models, 46
sevenbridges.models.actions, 55
sevenbridges.models.app, 55
sevenbridges.models.billing_breakdown, 56
sevenbridges.models.billing_group, 56
sevenbridges.models.compound, 46
sevenbridges.models.compound.billing, 46
sevenbridges.models.compound.billing.invoice_period, 46
sevenbridges.models.compound.billing.project_breakdown, 46
sevenbridges.models.compound.billing.task_breakdown, 47
sevenbridges.models.compound.error, 54
sevenbridges.models.compound.files, 47
sevenbridges.models.compound.files.download_info, 47
sevenbridges.models.compound.files.file_origin, 47
sevenbridges.models.compound.files.file_storage, 48
sevenbridges.models.compound.files.metadata, 48
sevenbridges.models.compound.jobs, 48
sevenbridges.models.compound.jobs.job, 48
sevenbridges.models.compound.jobs.job_docker, 49
sevenbridges.models.compound.jobs.job_instance, 49
sevenbridges.models.compound.jobs.job_instance_dis 49
sevenbridges.models.compound.jobs.job_log, 49
sevenbridges.models.compound.limits, 50
sevenbridges.models.compound.limits.rate, 50
sevenbridges.models.compound.markers, 50
sevenbridges.models.compound.markers.position, 50
sevenbridges.models.compound.price, 54
sevenbridges.models.compound.price_breakdown, 54
sevenbridges.models.compound.projects, 50
sevenbridges.models.compound.projects.permissions, 50
sevenbridges.models.compound.projects.settings, 50
sevenbridges.models.compound.tasks, 51
sevenbridges.models.compound.tasks.batch_by, 51
sevenbridges.models.compound.tasks.batch_group, 51
sevenbridges.models.compound.tasks.execution_statu 51
sevenbridges.models.compound.tasks.input, 52
sevenbridges.models.compound.tasks.output, 52

sevenbridges.models.compound.volumes, 52
sevenbridges.models.compound.volumes.import_destination, 52
sevenbridges.models.compound.volumes.properties, 53
sevenbridges.models.compound.volumes.service, 53
sevenbridges.models.compound.volumes.volume_file, 53
sevenbridges.models.compound.volumes.volume_object, 53
sevenbridges.models.compound.volumes.volume_prefix, 53
sevenbridges.models.division, 57
sevenbridges.models.endpoints, 57
sevenbridges.models.enums, 58
sevenbridges.models.execution_details, 59
sevenbridges.models.file, 60
sevenbridges.models.invoice, 62
sevenbridges.models.link, 62
sevenbridges.models.marker, 62
sevenbridges.models.member, 63
sevenbridges.models.project, 63
sevenbridges.models.rate_limit, 65
sevenbridges.models.storage_export, 66
sevenbridges.models.storage_import, 67
sevenbridges.models.task, 67
sevenbridges.models.team, 69
sevenbridges.models.team_member, 70
sevenbridges.models.user, 70
sevenbridges.models.volume, 71
sevenbridges.transfer, 72
sevenbridges.transfer.download, 72
sevenbridges.transfer.upload, 73
sevenbridges.transfer.utils, 74

Index

A

AAHeader (class in sevenbridges.http.client), 42
abort() (sevenbridges.models.task.Task method), 67
abort() (sevenbridges.Task method), 31
aborted (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 51
ABORTED (sevenbridges.models.enums.TaskStatus attribute), 58
ABORTED (sevenbridges.models.enums.TransferState attribute), 59
ABORTED (sevenbridges.TaskStatus attribute), 40
ABORTED (sevenbridges.TransferState attribute), 40
access_mode (sevenbridges.models.volume.Volume attribute), 71
access_mode (sevenbridges.Volume attribute), 36
account_limit (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 51
action_url (sevenbridges.Endpoints attribute), 29
action_url (sevenbridges.models.endpoints.Endpoints attribute), 57
Actions (class in sevenbridges.models.actions), 55
actions (sevenbridges.Api attribute), 26
actions (sevenbridges.api.Api attribute), 74
active (sevenbridges.models.volume.Volume attribute), 71
active (sevenbridges.Volume attribute), 36
add_callback() (sevenbridges.transfer.download.Download method), 72
add_callback() (sevenbridges.transfer.upload.Upload method), 73
add_error_handler() (sevenbridges.http.client.HttpClient method), 42
add_files() (sevenbridges.models.project.Project method), 63
add_files() (sevenbridges.Project method), 29
add_member() (sevenbridges.Dataset method), 39
add_member() (sevenbridges.models.project.Project method), 64
add_member() (sevenbridges.models.team.Team method), 69
add_member() (sevenbridges.models.volume.Volume method), 71
add_member() (sevenbridges.Project method), 29
add_member() (sevenbridges.Volume method), 37
add_member_division() (sevenbridges.models.project.Project method), 64
add_member_division() (sevenbridges.models.volume.Volume method), 71
add_member_division() (sevenbridges.Project method), 29
add_member_division() (sevenbridges.Volume method), 37
add_member_email() (sevenbridges.models.project.Project method), 64
add_member_email() (sevenbridges.Project method), 29
add_member_team() (sevenbridges.models.project.Project method), 64
add_member_team() (sevenbridges.models.volume.Volume method), 71
add_member_team() (sevenbridges.Project method), 29
add_member_team() (sevenbridges.Volume method), 37
add_progress_callback() (sevenbridges.transfer.download.Download method), 72
add_progress_callback() (sevenbridges.transfer.upload.Upload method), 73
address (sevenbridges.models.user.User attribute), 70
address (sevenbridges.User attribute), 28
advance_access (sevenbridges.config.Profile attribute), 75
AdvanceAccessError, 76
affiliation (sevenbridges.models.user.User attribute), 70
affiliation (sevenbridges.User attribute), 28
all() (sevenbridges.meta.collection.Collection method),

43
amount (sevenbridges.models.compound.price.Price attribute), 54
analysis_costs (sevenbridges.Invoice attribute), 27
analysis_costs (sevenbridges.models.invoice.Invoice attribute), 62
analysis_spending (sevenbridges.models.compound.billing.project_breakdown.ProjectBreakdown attribute), 46
Api (class in sevenbridges), 26
Api (class in sevenbridges.api), 74
api_endpoint (sevenbridges.config.Profile attribute), 75
App (class in sevenbridges), 32
App (class in sevenbridges.models.app), 55
app (sevenbridges.models.task.Task attribute), 68
app (sevenbridges.Task attribute), 31
AppRawFormat (class in sevenbridges.models.enums), 58
apps (sevenbridges.Api attribute), 26
apps (sevenbridges.api.Api attribute), 74
apps_url (sevenbridges.Endpoints attribute), 29
apps_url (sevenbridges.models.endpoints.Endpoints attribute), 57
auth_token (sevenbridges.config.Profile attribute), 75

B

BadRequest, 41, 76
balance (sevenbridges.BillingGroup attribute), 27
balance (sevenbridges.models.billing_group.BillingGroup attribute), 56
bandwidth (sevenbridges.transfer.utils.Progress attribute), 74
BasicListField (class in sevenbridges.meta.fields), 44
batch (sevenbridges.models.task.Task attribute), 68
batch (sevenbridges.Task attribute), 31
batch_by (sevenbridges.models.task.Task attribute), 68
batch_by (sevenbridges.Task attribute), 31
batch_group (sevenbridges.models.task.Task attribute), 68
batch_group (sevenbridges.Task attribute), 31
batch_input (sevenbridges.models.task.Task attribute), 68
batch_input (sevenbridges.Task attribute), 31
BatchBy (class in sevenbridges.models.compound.tasks.batch_by), 51
BatchGroup (class in sevenbridges.models.compound.tasks.batch_group), 51
billing_group (sevenbridges.models.project.Project attribute), 64
billing_group (sevenbridges.Project attribute), 29
billing_groups (sevenbridges.Api attribute), 26
billing_groups (sevenbridges.api.Api attribute), 74
billing_url (sevenbridges.Endpoints attribute), 29
billing_url (sevenbridges.models.endpoints.Endpoints attribute), 57
BillingGroup (class in sevenbridges), 27
BillingGroup (class in sevenbridges.models.billing_group), 56
BillingGroupBreakdown (class in sevenbridges), 28
BillingGroupBreakdown (class in sevenbridges.models.billing_breakdown), 56
BooleanField (class in sevenbridges.meta.fields), 44
Breakdown (class in sevenbridges.models.compound.price_breakdown), 54
breakdown (sevenbridges.models.compound.price.Price attribute), 54
breakdown() (sevenbridges.BillingGroup method), 27
breakdown() (sevenbridges.models.billing_group.BillingGroup method), 56
bulk_copy_files() (sevenbridges.models.actions.Actions class method), 55
bulk_delete() (sevenbridges.File class method), 33
bulk_delete() (sevenbridges.models.file.File class method), 60
bulk_edit() (sevenbridges.File class method), 33
bulk_edit() (sevenbridges.models.file.File class method), 60
bulk_get() (sevenbridges.Export class method), 35
bulk_get() (sevenbridges.File class method), 33
bulk_get() (sevenbridges.Import class method), 36
bulk_get() (sevenbridges.models.file.File class method), 60
bulk_get() (sevenbridges.models.storage_export.Export class method), 66
bulk_get() (sevenbridges.models.storage_import.Import class method), 67
bulk_submit() (sevenbridges.Export class method), 35
bulk_submit() (sevenbridges.Import class method), 36
bulk_submit() (sevenbridges.models.storage_export.Export class method), 66
bulk_submit() (sevenbridges.models.storage_import.Import class method), 67
bulk_update() (sevenbridges.File class method), 33
bulk_update() (sevenbridges.models.file.File class method), 60
BulkRecord (class in sevenbridges), 39
bytes_done (sevenbridges.transfer.utils.Progress attribute), 74

C

check_for_error() (in module sevenbridges.decorators), 76
checksum (sevenbridges.models.compound.jobs.job_docker.JobDocker attribute), 49
chromosome (sevenbridges.Marker attribute), 38

chromosome (sevenbridges.models.marker.Marker attribute), 62
city (sevenbridges.models.user.User attribute), 70
city (sevenbridges.User attribute), 28
code (sevenbridges.models.compound.error.Error attribute), 54
Collection (class in sevenbridges.meta.collection), 43
command_line (sevenbridges.models.compound.jobs.job.Job attribute), 48
COMPLETED (sevenbridges.ImportExportState attribute), 40
completed (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 51
COMPLETED (sevenbridges.models.enums.ImportExportState attribute), 58
COMPLETED (sevenbridges.models.enums.TaskStatus attribute), 58
COMPLETED (sevenbridges.models.enums.TransferState attribute), 59
COMPLETED (sevenbridges.TaskStatus attribute), 40
COMPLETED (sevenbridges.TransferState attribute), 40
CompoundField (class in sevenbridges.meta.fields), 44
CompoundListField (class in sevenbridges.meta.fields), 44
CompoundMutableDict (class in sevenbridges.meta.comp_mutable_dict), 43
computation (sevenbridges.models.compound.price_breakdown.Breakdown attribute), 54
Config (class in sevenbridges), 27
Config (class in sevenbridges.config), 75
CONFIG (sevenbridges.config.Profile attribute), 75
config_vars() (in module sevenbridges.http.client), 42
Conflict, 41, 76
content() (sevenbridges.File method), 33
content() (sevenbridges.models.file.File method), 60
copy() (sevenbridges.App method), 32
copy() (sevenbridges.File method), 34
copy() (sevenbridges.models.app.App method), 55
copy() (sevenbridges.models.file.File method), 60
country (sevenbridges.models.user.User attribute), 70
country (sevenbridges.User attribute), 28
create() (sevenbridges.Marker class method), 38
create() (sevenbridges.models.marker.Marker class method), 62
create() (sevenbridges.models.project.Project class method), 64
create() (sevenbridges.models.task.Task class method), 68
create() (sevenbridges.models.team.Team class method), 69
create() (sevenbridges.Project class method), 29
create() (sevenbridges.Task class method), 31
create_google_volume() (sevenbridges.models.volume.Volume class method), 71
create_google_volume() (sevenbridges.Volume class method), 37
create_revision() (sevenbridges.App class method), 32
create_revision() (sevenbridges.models.app.App class method), 55
create_s3_volume() (sevenbridges.models.volume.Volume class method), 71
create_s3_volume() (sevenbridges.Volume class method), 37
create_task() (sevenbridges.models.project.Project method), 64
create_task() (sevenbridges.Project method), 29
created_by (sevenbridges.Marker attribute), 38
created_by (sevenbridges.models.marker.Marker attribute), 63
created_by (sevenbridges.models.task.Task attribute), 68
created_by (sevenbridges.Task attribute), 31
created_on (sevenbridges.File attribute), 34
created_on (sevenbridges.models.file.File attribute), 60
created_on (sevenbridges.models.volume.Volume attribute), 71
created_on (sevenbridges.Volume attribute), 37
created_time (sevenbridges.Marker attribute), 38
created_time (sevenbridges.models.marker.Marker attribute), 63
created_time (sevenbridges.models.task.Task attribute), 68
created_time (sevenbridges.Task attribute), 31
CREATING (sevenbridges.models.enums.TaskStatus attribute), 58
CREATING (sevenbridges.TaskStatus attribute), 40
CREDENTIALS (sevenbridges.config.Profile attribute), 75
currency (sevenbridges.models.compound.price.Price attribute), 54

D

DataContainer (class in sevenbridges.meta.data), 43
Dataset (class in sevenbridges), 39
datasets (sevenbridges.Api attribute), 26
datasets (sevenbridges.api.Api attribute), 75
DateTimeField (class in sevenbridges.meta.fields), 44
deepcopy() (sevenbridges.App method), 32
deepcopy() (sevenbridges.BillingGroup method), 27
deepcopy() (sevenbridges.BillingGroupBreakdown method), 28
deepcopy() (sevenbridges.BulkRecord method), 39
deepcopy() (sevenbridges.Dataset method), 39
deepcopy() (sevenbridges.Division method), 38
deepcopy() (sevenbridges.Endpoints method), 29
deepcopy() (sevenbridges.Export method), 35
deepcopy() (sevenbridges.File method), 34
deepcopy() (sevenbridges.Import method), 36

deepcopy() (sevenbridges.Invoice method), 27
deepcopy() (sevenbridges.Marker method), 38
deepcopy() (sevenbridges.Member method), 33
deepcopy() (sevenbridges.models.actions.Actions method), 55
deepcopy() (sevenbridges.models.app.App method), 55
deepcopy() (sevenbridges.models.billing_breakdown.BillingGroupBreakdown method), 56
deepcopy() (sevenbridges.models.billing_group.BillingGroup method), 56
deepcopy() (sevenbridges.models.compound.billing.invoice_period.InvoicePeriod method), 46
deepcopy() (sevenbridges.models.compound.billing.project_breakdown.ProjectBreakdown method), 46
deepcopy() (sevenbridges.models.compound.billing.task_breakdown.TaskBreakdown method), 47
deepcopy() (sevenbridges.models.compound.error.Error method), 54
deepcopy() (sevenbridges.models.compound.files.download_info.DownloadInfo method), 47
deepcopy() (sevenbridges.models.compound.files.file_origin.FileOrigin method), 47
deepcopy() (sevenbridges.models.compound.files.file_storage.FileStorage method), 48
deepcopy() (sevenbridges.models.compound.jobs.job.Job method), 48
deepcopy() (sevenbridges.models.compound.jobs.job_docker.JobDocker method), 49
deepcopy() (sevenbridges.models.compound.jobs.job_instance.Instance method), 49
deepcopy() (sevenbridges.models.compound.jobs.job_instance_disk.Disk method), 49
deepcopy() (sevenbridges.models.compound.limits.rate.Rate method), 50
deepcopy() (sevenbridges.models.compound.price.Price method), 54
deepcopy() (sevenbridges.models.compound.price_breakdown.Breakdown method), 54
deepcopy() (sevenbridges.models.compound.tasks.batch_group.BatchGroup method), 51
deepcopy() (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus method), 51
deepcopy() (sevenbridges.models.compound.volumes.import_destination.ImportDestination method), 52
deepcopy() (sevenbridges.models.compound.volumes.volume_file.VolumeFile method), 53
deepcopy() (sevenbridges.models.compound.volumes.volume_object.VolumeObject method), 53
deepcopy() (sevenbridges.models.compound.volumes.volume_prefix.VolumePrefix method), 53
deepcopy() (sevenbridges.models.division.Division method), 57
deepcopy() (sevenbridges.models.endpoints.Endpoints method), 57
deepcopy() (sevenbridges.models.execution_details.ExecutionDetails method), 59
deepcopy() (sevenbridges.models.file.File method), 60
deepcopy() (sevenbridges.models.file.FileBulkRecord method), 61
deepcopy() (sevenbridges.models.invoice.Invoice method), 62
deepcopy() (sevenbridges.models.link.Link method), 62
deepcopy() (sevenbridges.models.link.VolumeLink method), 62
deepcopy() (sevenbridges.models.marker.Marker method), 63
deepcopy() (sevenbridges.models.member.Member method), 63
deepcopy() (sevenbridges.models.project.Project method), 64
deepcopy() (sevenbridges.models.rate_limit.RateLimit method), 65
deepcopy() (sevenbridges.models.storage_export.Export method), 66
deepcopy() (sevenbridges.models.storage_export.ExportBulkRecord method), 66
deepcopy() (sevenbridges.models.storage_import.Import method), 67
deepcopy() (sevenbridges.models.storage_import.ImportBulkRecord method), 67
deepcopy() (sevenbridges.models.task.Task method), 68
deepcopy() (sevenbridges.models.team.Team method), 69
deepcopy() (sevenbridges.models.team_member.TeamMember method), 70
deepcopy() (sevenbridges.models.user.User method), 70
deepcopy() (sevenbridges.models.volume.Volume method), 71
deepcopy() (sevenbridges.Project method), 30
deepcopy() (sevenbridges.Task method), 31
deepcopy() (sevenbridges.User method), 28
deepcopy() (sevenbridges.Volume method), 37
delete() (sevenbridges.http.client.HttpClient method), 42
delete() (sevenbridges.Marker method), 38
delete() (sevenbridges.meta.resource.Resource method), 45
delete() (sevenbridges.models.marker.Marker method), 63
delete() (sevenbridges.models.team.Team method), 69
description (sevenbridges.Dataset attribute), 39
description (sevenbridges.models.project.Project attribute), 64
description (sevenbridges.models.task.Task attribute), 68
description (sevenbridges.models.volume.Volume attribute), 71
description (sevenbridges.Project attribute), 30
description (sevenbridges.Task attribute), 31
description (sevenbridges.Volume attribute), 37
destination (sevenbridges.Export attribute), 35

destination (sevenbridges.Import attribute), 36
destination (sevenbridges.models.storage_export.Export attribute), 66
destination (sevenbridges.models.storage_import.Import attribute), 67
DictField (class in sevenbridges.meta.fields), 44
disabled (sevenbridges.BillingGroup attribute), 27
disabled (sevenbridges.models.billing_group.BillingGroup attribute), 56
Disk (class in sevenbridges.models.compound.jobs.job_instance_disk), 49
disk (sevenbridges.models.compound.jobs.job_instance.Instance attribute), 49
Division (class in sevenbridges), 38
Division (class in sevenbridges.models.division), 57
divisions (sevenbridges.Api attribute), 26
divisions (sevenbridges.api.Api attribute), 75
docker (sevenbridges.models.compound.jobs.job.Job attribute), 48
done() (sevenbridges.transfer.download.DPartedFile method), 72
done() (sevenbridges.transfer.upload.UPartedFile method), 73
Download (class in sevenbridges.transfer.download), 72
download() (sevenbridges.File method), 34
download() (sevenbridges.models.file.File method), 60
download_info() (sevenbridges.File method), 34
download_info() (sevenbridges.models.file.File method), 60
DOWNLOAD_MINIMUM_PART_SIZE (sevenbridges.models.enums.PartSize attribute), 58
DownloadInfo (class in sevenbridges.models.compound.files.download_info), 47
DPartedFile (class in sevenbridges.transfer.download), 72
DRAFT (sevenbridges.models.enums.TaskStatus attribute), 58
DRAFT (sevenbridges.TaskStatus attribute), 40
duration (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 51
duration (sevenbridges.transfer.download.Download attribute), 72
duration (sevenbridges.transfer.upload.Upload attribute), 73
duration (sevenbridges.transfer.utils.Progress attribute), 74

E

email (sevenbridges.Member attribute), 33
email (sevenbridges.models.member.Member attribute), 63
email (sevenbridges.models.user.User attribute), 70
email (sevenbridges.User attribute), 28
end_time (sevenbridges.models.compound.jobs.job.Job attribute), 48
end_time (sevenbridges.models.execution_details.ExecutionDetails attribute), 59
end_time (sevenbridges.models.task.Task attribute), 68
end_time (sevenbridges.Task attribute), 31
Endpoints (class in sevenbridges), 28
Endpoints (class in sevenbridges.models.endpoints), 57
endpoints (sevenbridges.Api attribute), 26
endpoints (sevenbridges.api.Api attribute), 75
equals() (sevenbridges.App method), 32
equals() (sevenbridges.BillingGroup method), 27
equals() (sevenbridges.BillingGroupBreakdown method), 28
equals() (sevenbridges.BulkRecord method), 39
equals() (sevenbridges.Dataset method), 39
equals() (sevenbridges.Division method), 38
equals() (sevenbridges.Endpoints method), 29
equals() (sevenbridges.Export method), 35
equals() (sevenbridges.File method), 34
equals() (sevenbridges.Import method), 36
equals() (sevenbridges.Invoice method), 27
equals() (sevenbridges.Marker method), 38
equals() (sevenbridges.Member method), 33
equals() (sevenbridges.meta.comp_mutable_dict.CompoundMutableDict method), 43
equals() (sevenbridges.models.actions.Actions method), 55
equals() (sevenbridges.models.app.App method), 55
equals() (sevenbridges.models.billing_breakdown.BillingGroupBreakdown method), 56
equals() (sevenbridges.models.billing_group.BillingGroup method), 56
equals() (sevenbridges.models.compound.billing.invoice_period.InvoicePeri method), 46
equals() (sevenbridges.models.compound.billing.project_breakdown.Projec method), 46
equals() (sevenbridges.models.compound.billing.task_breakdown.TaskBrea method), 47
equals() (sevenbridges.models.compound.error.Error method), 54
equals() (sevenbridges.models.compound.files.download_info.DownloadInf method), 47
equals() (sevenbridges.models.compound.files.file_origin.FileOrigin method), 47
equals() (sevenbridges.models.compound.files.file_storage.FileStorage method), 48
equals() (sevenbridges.models.compound.jobs.job.Job method), 48
equals() (sevenbridges.models.compound.jobs.job_docker.JobDocker method), 49
equals() (sevenbridges.models.compound.jobs.job_instance.Instance method), 49

equals() (sevenbridges.models.compound.jobs.job_instance_disk.Disk method), 49
equals() (sevenbridges.models.compound.limits.rate.Rate method), 50
equals() (sevenbridges.models.compound.price.Price method), 54
equals() (sevenbridges.models.compound.price_breakdown.Breakdown method), 54
equals() (sevenbridges.models.compound.tasks.batch_by.BatchBy method), 51
equals() (sevenbridges.models.compound.tasks.batch_group.BatchGroup method), 51
equals() (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus method), 51
equals() (sevenbridges.models.compound.volumes.import_destination.ImportDestination method), 52
equals() (sevenbridges.models.compound.volumes.volume_file.VolumeFile method), 53
equals() (sevenbridges.models.compound.volumes.volume_object.VolumeObject method), 53
equals() (sevenbridges.models.compound.volumes.volume_prefix.VolumePrefix method), 54
equals() (sevenbridges.models.division.Division method), 57
equals() (sevenbridges.models.endpoints.Endpoints method), 57
equals() (sevenbridges.models.execution_details.ExecutionDetails method), 59
equals() (sevenbridges.models.file.File method), 60
equals() (sevenbridges.models.file.FileBulkRecord method), 61
equals() (sevenbridges.models.invoice.Invoice method), 62
equals() (sevenbridges.models.link.Link method), 62
equals() (sevenbridges.models.link.VolumeLink method), 62
equals() (sevenbridges.models.marker.Marker method), 63
equals() (sevenbridges.models.member.Member method), 63
equals() (sevenbridges.models.project.Project method), 64
equals() (sevenbridges.models.rate_limit.RateLimit method), 65
equals() (sevenbridges.models.storage_export.Export method), 66
equals() (sevenbridges.models.storage_export.ExportBulkRecord method), 66
equals() (sevenbridges.models.storage_import.Import method), 67
equals() (sevenbridges.models.storage_import.ImportBulkRecord method), 67
equals() (sevenbridges.models.task.Task method), 68
equals() (sevenbridges.models.team.Team method), 69
equals() (sevenbridges.models.team_member.TeamMember method), 70
equals() (sevenbridges.models.user.User method), 70
equals() (sevenbridges.models.volume.Volume method), 71
equals() (sevenbridges.Project method), 30
equals() (sevenbridges.Task method), 31
equals() (sevenbridges.User method), 28
equals() (sevenbridges.Volume method), 37
Error (class in sevenbridges.models.compound.error), 54
error (sevenbridges.BulkRecord attribute), 39
error (sevenbridges.Export attribute), 35
error (sevenbridges.Import attribute), 36
error (sevenbridges.models.storage_export.Export attribute), 66
error (sevenbridges.models.storage_import.Import attribute), 67
errors (sevenbridges.models.task.Task attribute), 68
errors (sevenbridges.Task attribute), 31
executed_by (sevenbridges.models.task.Task attribute), 68
executed_by (sevenbridges.Task attribute), 31
execution_duration (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 51
execution_status (sevenbridges.models.task.Task attribute), 68
execution_status (sevenbridges.Task attribute), 31
ExecutionDetails (class in sevenbridges.models.execution_details), 59
ExecutionDetailsInvalidTaskType, 41, 76
ExecutionStatus (class in sevenbridges.models.compound.tasks.execution_status), 51
Export (class in sevenbridges), 35
Export (class in sevenbridges.models.storage_export), 66
ExportBulkRecord (class in sevenbridges.models.storage_export), 66
exports (sevenbridges.Api attribute), 26
exports (sevenbridges.api.Api attribute), 75

F

FAILED (sevenbridges.ImportExportState attribute), 40
failed (sevenbridges.models.compound.tasks.execution_status.ExecutionSta attribute), 52
FAILED (sevenbridges.models.enums.ImportExportState attribute), 58
FAILED (sevenbridges.models.enums.TaskStatus attribute), 59
FAILED (sevenbridges.models.enums.TransferState attribute), 59
FAILED (sevenbridges.TaskStatus attribute), 40
FAILED (sevenbridges.TransferState attribute), 40
FeedbackType (class in sevenbridges.models.enums), 58

fetch() (sevenbridges.meta.data.DataContainer method), 43
Field (class in sevenbridges.meta.fields), 44
fields (sevenbridges.models.compound.tasks.batch_group.BatchGroup attribute), 51
File (class in sevenbridges), 33
File (class in sevenbridges.models.file), 60
file (sevenbridges.Marker attribute), 38
file (sevenbridges.models.marker.Marker attribute), 63
file_name (sevenbridges.transfer.upload.Upload attribute), 73
file_size (sevenbridges.transfer.utils.Progress attribute), 74
FileBulkRecord (class in sevenbridges.models.file), 61
FileOrigin (class in sevenbridges.models.compound.files.file_origin), 47
files (sevenbridges.Api attribute), 26
files (sevenbridges.api.Api attribute), 75
files_url (sevenbridges.Endpoints attribute), 29
files_url (sevenbridges.models.endpoints.Endpoints attribute), 57
FileStorage (class in sevenbridges.models.compound.files.file_storage), 48
FileStorageType (class in sevenbridges), 40
FileStorageType (class in sevenbridges.models.enums), 58
finished_on (sevenbridges.Export attribute), 35
finished_on (sevenbridges.Import attribute), 36
finished_on (sevenbridges.models.storage_export.Export attribute), 66
finished_on (sevenbridges.models.storage_import.Import attribute), 67
first_name (sevenbridges.models.user.User attribute), 70
first_name (sevenbridges.User attribute), 28
FloatField (class in sevenbridges.meta.fields), 44
Forbidden, 41, 76
format_proxies() (in module sevenbridges.config), 76
from_ (sevenbridges.models.compound.billing.invoice_period.InvoicePeriod attribute), 46

G

GB (sevenbridges.models.enums.PartSize attribute), 58
general_error_sleeper() (in module sevenbridges.http.error_handlers), 42
generate_session() (in module sevenbridges.http.client), 42
get() (sevenbridges.Division class method), 38
get() (sevenbridges.Endpoints class method), 29
get() (sevenbridges.http.client.HttpClient method), 42
get() (sevenbridges.meta.resource.Resource class method), 45
get() (sevenbridges.models.division.Division class method), 57
get() (sevenbridges.models.endpoints.Endpoints class method), 57
get() (sevenbridges.models.rate_limit.RateLimit class method), 65
get() (sevenbridges.models.team.Team class method), 69
get() (sevenbridges.models.user.User class method), 70
get() (sevenbridges.User class method), 28
get_apps() (sevenbridges.models.project.Project method), 64
get_apps() (sevenbridges.Project method), 30
get_batch_children() (sevenbridges.models.task.Task method), 68
get_batch_children() (sevenbridges.Task method), 31
get_execution_details() (sevenbridges.models.task.Task method), 68
get_execution_details() (sevenbridges.Task method), 31
get_exports() (sevenbridges.models.project.Project method), 64
get_exports() (sevenbridges.models.volume.Volume method), 71
get_exports() (sevenbridges.Project method), 30
get_exports() (sevenbridges.Volume method), 37
get_files() (sevenbridges.models.project.Project method), 65
get_files() (sevenbridges.Project method), 30
get_imports() (sevenbridges.models.project.Project method), 65
get_imports() (sevenbridges.models.volume.Volume method), 71
get_imports() (sevenbridges.Project method), 30
get_imports() (sevenbridges.Volume method), 37
get_member() (sevenbridges.Dataset method), 39
get_members() (sevenbridges.Dataset method), 39
get_members() (sevenbridges.models.project.Project method), 65
get_members() (sevenbridges.models.team.Team method), 69
get_members() (sevenbridges.models.volume.Volume method), 71
get_members() (sevenbridges.Project method), 30
get_members() (sevenbridges.Volume method), 37
get_owned_by() (sevenbridges.Dataset class method), 39
get_parts() (sevenbridges.transfer.download.DPartedFile method), 72
get_parts() (sevenbridges.transfer.upload.UPartedFile method), 73
get_revision() (sevenbridges.App class method), 32
get_revision() (sevenbridges.models.app.App class method), 55
get_tasks() (sevenbridges.models.project.Project method), 65
get_tasks() (sevenbridges.Project method), 30

get_teams() (sevenbridges.Division method), 38
get_teams() (sevenbridges.models.division.Division method), 57
get_volume_object_info() (sevenbridges.models.volume.Volume method), 71
get_volume_object_info() (sevenbridges.Volume method), 37
GOOGLE (sevenbridges.models.enums.VolumeType attribute), 59
GOOGLE (sevenbridges.VolumeType attribute), 40

H

href (sevenbridges.App attribute), 32
href (sevenbridges.BillingGroup attribute), 28
href (sevenbridges.BillingGroupBreakdown attribute), 28
href (sevenbridges.Dataset attribute), 39
href (sevenbridges.Division attribute), 38
href (sevenbridges.Export attribute), 35
href (sevenbridges.File attribute), 34
href (sevenbridges.Import attribute), 36
href (sevenbridges.Invoice attribute), 27
href (sevenbridges.Marker attribute), 38
href (sevenbridges.Member attribute), 33
href (sevenbridges.models.app.App attribute), 55
href (sevenbridges.models.billing_breakdown.BillingGroupBreakdown attribute), 56
href (sevenbridges.models.billing_group.BillingGroup attribute), 56
href (sevenbridges.models.compound.billing.project_breakdown.ProjectBreakdown attribute), 46
href (sevenbridges.models.compound.billing.task_breakdown.TaskBreakdown attribute), 47
href (sevenbridges.models.compound.volumes.volume_object.VolumeObject attribute), 53
href (sevenbridges.models.compound.volumes.volume_prefix.VolumePrefix attribute), 54
href (sevenbridges.models.division.Division attribute), 57
href (sevenbridges.models.execution_details.ExecutionDetails attribute), 59
href (sevenbridges.models.file.File attribute), 60
href (sevenbridges.models.invoice.Invoice attribute), 62
href (sevenbridges.models.link.Link attribute), 62
href (sevenbridges.models.marker.Marker attribute), 63
href (sevenbridges.models.member.Member attribute), 63
href (sevenbridges.models.project.Project attribute), 65
href (sevenbridges.models.storage_export.Export attribute), 66
href (sevenbridges.models.storage_import.Import attribute), 67
href (sevenbridges.models.task.Task attribute), 68
href (sevenbridges.models.team.Team attribute), 69
href (sevenbridges.models.team_member.TeamMember attribute), 70
href (sevenbridges.models.user.User attribute), 70
href (sevenbridges.models.volume.Volume attribute), 71
href (sevenbridges.Project attribute), 30
href (sevenbridges.Task attribute), 31
href (sevenbridges.User attribute), 28
href (sevenbridges.Volume attribute), 37
HrefField (class in sevenbridges.meta.fields), 44
HttpClient (class in sevenbridges.http.client), 42

I

id (sevenbridges.App attribute), 32
id (sevenbridges.BillingGroup attribute), 28
id (sevenbridges.Dataset attribute), 39
id (sevenbridges.Division attribute), 38
id (sevenbridges.Export attribute), 35
id (sevenbridges.File attribute), 34
id (sevenbridges.Import attribute), 36
id (sevenbridges.Invoice attribute), 27
id (sevenbridges.Marker attribute), 38
id (sevenbridges.Member attribute), 33
id (sevenbridges.models.app.App attribute), 55
id (sevenbridges.models.billing_group.BillingGroup attribute), 56
id (sevenbridges.models.compound.jobs.job_instance.Instance attribute), 49
id (sevenbridges.models.division.Division attribute), 57
id (sevenbridges.models.file.File attribute), 60
id (sevenbridges.models.invoice.Invoice attribute), 62
id (sevenbridges.models.marker.Marker attribute), 63
id (sevenbridges.models.member.Member attribute), 63
id (sevenbridges.models.project.Project attribute), 65
id (sevenbridges.models.storage_export.Export attribute), 66
id (sevenbridges.models.storage_import.Import attribute), 67
id (sevenbridges.models.task.Task attribute), 68
id (sevenbridges.models.team.Team attribute), 69
id (sevenbridges.models.team_member.TeamMember attribute), 70
id (sevenbridges.models.volume.Volume attribute), 71
id (sevenbridges.Project attribute), 30
id (sevenbridges.Task attribute), 31
id (sevenbridges.Volume attribute), 37
IDEA (sevenbridges.models.enums.FeedbackType attribute), 58
Import (class in sevenbridges), 36
Import (class in sevenbridges.models.storage_import), 67
ImportBulkRecord (class in sevenbridges.models.storage_import), 67
ImportDestination (class in sevenbridges.models.compound.volumes.import_destination), 52
ImportExportState (class in sevenbridges), 40

ImportExportState (class in sevenbridges.models.enums), 58
imports (sevenbridges.Api attribute), 26
imports (sevenbridges.api.Api attribute), 75
inplace_reload() (in module sevenbridges.decorators), 76
Input (class in sevenbridges.models.compound.tasks.input), 52
inputs (sevenbridges.models.task.Task attribute), 68
inputs (sevenbridges.Task attribute), 31
install_app() (sevenbridges.App class method), 32
install_app() (sevenbridges.models.app.App class method), 55
Instance (class in sevenbridges.models.compound.jobs.job_instance), 49
instance (sevenbridges.models.compound.jobs.job.Job attribute), 48
instance_init (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 52
instance_limit (sevenbridges.models.rate_limit.RateLimit attribute), 65
IntegerField (class in sevenbridges.meta.fields), 44
Invoice (class in sevenbridges), 27
Invoice (class in sevenbridges.models.invoice), 62
invoice_period (sevenbridges.Invoice attribute), 27
invoice_period (sevenbridges.models.invoice.Invoice attribute), 62
InvoicePeriod (class in sevenbridges.models.compound.billing.invoice_period), 46
invoices (sevenbridges.Api attribute), 26
invoices (sevenbridges.api.Api attribute), 75
items() (sevenbridges.meta.comp_mutable_dict.CompoundMutableDict method), 43

J

Job (class in sevenbridges.models.compound.jobs.job), 48
JobDocker (class in sevenbridges.models.compound.jobs.job_docker), 49
jobs (sevenbridges.models.execution_details.ExecutionDetails attribute), 59
JSON (sevenbridges.models.enums.AppRawFormat attribute), 58

K

KB (sevenbridges.models.enums.PartSize attribute), 58
key (sevenbridges.http.client.AAHeader attribute), 42

L

last_name (sevenbridges.models.user.User attribute), 70
last_name (sevenbridges.User attribute), 28
limit (sevenbridges.http.client.HttpClient attribute), 42
limit (sevenbridges.models.compound.limits.rate.Rate attribute), 50
Link (class in sevenbridges.models.link), 62
list() (sevenbridges.models.volume.Volume method), 72
list() (sevenbridges.Volume method), 37
LocalFileAlreadyExists, 41, 76
location (sevenbridges.models.compound.files.file_storage.FileStorage attribute), 48
location (sevenbridges.models.compound.volumes.volume_file.VolumeFile attribute), 53
location (sevenbridges.models.compound.volumes.volume_object.VolumeC attribute), 53
Logs (class in sevenbridges.models.compound.jobs.job_log), 49
logs (sevenbridges.models.compound.jobs.job.Job attribute), 48

M

maintenance_sleeper() (in module sevenbridges.http.error_handlers), 42
map_input_output() (in module sevenbridges.models.compound.tasks), 51
Marker (class in sevenbridges), 38
Marker (class in sevenbridges.models.marker), 62
MarkerPosition (class in sevenbridges.models.compound.markers.position), 50
markers (sevenbridges.Api attribute), 26
markers (sevenbridges.api.Api attribute), 75
MAXIMUM_OBJECT_SIZE (sevenbridges.models.enums.PartSize attribute), 58
MAXIMUM_TOTAL_PARTS (sevenbridges.models.enums.PartSize attribute), 58
MAXIMUM_UPLOAD_SIZE (sevenbridges.models.enums.PartSize attribute), 58
MB (sevenbridges.models.enums.PartSize attribute), 58
me() (sevenbridges.models.user.User class method), 70
me() (sevenbridges.User class method), 28
Member (class in sevenbridges), 33
Member (class in sevenbridges.models.member), 63
message (sevenbridges.models.compound.error.Error attribute), 54
message (sevenbridges.models.compound.tasks.execution_status.Execution attribute), 52
message (sevenbridges.models.execution_details.ExecutionDetails attribute), 59
message_code (sevenbridges.models.compound.tasks.execution_status.Exec attribute), 52
Metadata (class in sevenbridges.models.compound.files.metadata),

48
metadata (sevenbridges.File attribute), 34
metadata (sevenbridges.models.compound.volumes.volume_object.VolumeObject attribute), 53
metadata (sevenbridges.models.file.File attribute), 60
method (sevenbridges.models.link.Link attribute), 62
MethodNotAllowed, 41, 76
modified_on (sevenbridges.File attribute), 34
modified_on (sevenbridges.models.file.File attribute), 61
modified_on (sevenbridges.models.volume.Volume attribute), 72
modified_on (sevenbridges.Volume attribute), 37
more_info (sevenbridges.models.compound.error.Error attribute), 54

N

name (sevenbridges.App attribute), 32
name (sevenbridges.BillingGroup attribute), 28
name (sevenbridges.Dataset attribute), 39
name (sevenbridges.Division attribute), 38
name (sevenbridges.File attribute), 34
name (sevenbridges.Marker attribute), 38
name (sevenbridges.models.app.App attribute), 55
name (sevenbridges.models.billing_group.BillingGroup attribute), 56
name (sevenbridges.models.compound.jobs.job.Job attribute), 48
name (sevenbridges.models.compound.volumes.import_destination.ImportDestination attribute), 52
name (sevenbridges.models.division.Division attribute), 57
name (sevenbridges.models.file.File attribute), 61
name (sevenbridges.models.marker.Marker attribute), 63
name (sevenbridges.models.project.Project attribute), 65
name (sevenbridges.models.task.Task attribute), 68
name (sevenbridges.models.team.Team attribute), 69
name (sevenbridges.models.volume.Volume attribute), 72
name (sevenbridges.Project attribute), 30
name (sevenbridges.Task attribute), 32
name (sevenbridges.Volume attribute), 37
next (sevenbridges.models.link.VolumeLink attribute), 62
next_page() (sevenbridges.meta.collection.Collection method), 43
next_page() (sevenbridges.meta.collection.VolumeCollection method), 43
NotFound, 41, 76
num_of_parts (sevenbridges.transfer.utils.Progress attribute), 74

O

ObjectIdField (class in sevenbridges.meta.fields), 44
origin (sevenbridges.File attribute), 34
origin (sevenbridges.models.file.File attribute), 61
Output (class in sevenbridges.models.compound.tasks.output), 52
outputs (sevenbridges.models.task.Task attribute), 68
outputs (sevenbridges.Task attribute), 32
overwrite (sevenbridges.Export attribute), 35
overwrite (sevenbridges.Import attribute), 36
overwrite (sevenbridges.models.storage_export.Export attribute), 66
overwrite (sevenbridges.models.storage_import.Import attribute), 67
owner (sevenbridges.BillingGroup attribute), 28
owner (sevenbridges.models.billing_group.BillingGroup attribute), 56

P

PaginationError, 41, 76
parent (sevenbridges.models.task.Task attribute), 68
parent (sevenbridges.Task attribute), 32
parse_records() (sevenbridges.BulkRecord class method), 39
Part (class in sevenbridges.transfer.utils), 74
parts_done (sevenbridges.transfer.utils.Progress attribute), 74
PartSize (class in sevenbridges.models.enums), 58
patch() (sevenbridges.http.client.HttpClient method), 42
path (sevenbridges.transfer.download.Download attribute), 72
pause() (sevenbridges.transfer.download.Download method), 72
pause() (sevenbridges.transfer.upload.Upload method), 73
PAUSED (sevenbridges.models.enums.TransferState attribute), 59
PAUSED (sevenbridges.TransferState attribute), 40
pending (sevenbridges.BillingGroup attribute), 28
PENDING (sevenbridges.ImportExportState attribute), 40
pending (sevenbridges.Invoice attribute), 27
pending (sevenbridges.models.billing_group.BillingGroup attribute), 56
PENDING (sevenbridges.models.enums.ImportExportState attribute), 58
pending (sevenbridges.models.invoice.Invoice attribute), 62
Permissions (class in sevenbridges), 33
Permissions (class in sevenbridges.models.compound.projects.permissions), 50
permissions (sevenbridges.Member attribute), 33
permissions (sevenbridges.models.member.Member attribute), 63
phone (sevenbridges.models.user.User attribute), 70
phone (sevenbridges.User attribute), 28

PLATFORM (sevenbridges.FileStorageType attribute), 40
PLATFORM (sevenbridges.models.enums.FileStorageType attribute), 58
position (sevenbridges.Marker attribute), 38
position (sevenbridges.models.marker.Marker attribute), 63
post() (sevenbridges.http.client.HttpClient method), 42
prefix (sevenbridges.models.compound.volumes.volume_prefix.VolumePrefix attribute), 54
PREPARING (sevenbridges.models.enums.TransferState attribute), 59
PREPARING (sevenbridges.TransferState attribute), 40
previous_page() (sevenbridges.meta.collection.Collection method), 43
previous_page() (sevenbridges.meta.collection.VolumeCollection method), 43
Price (class in sevenbridges.models.compound.price), 54
price (sevenbridges.models.task.Task attribute), 68
price (sevenbridges.Task attribute), 32
PROBLEM (sevenbridges.models.enums.FeedbackType attribute), 58
Profile (class in sevenbridges.config), 75
Progress (class in sevenbridges.transfer.utils), 74
progress (sevenbridges.transfer.download.Download attribute), 72
progress (sevenbridges.transfer.upload.Upload attribute), 73
progress (sevenbridges.transfer.utils.Progress attribute), 74
Project (class in sevenbridges), 29
Project (class in sevenbridges.models.project), 63
project (sevenbridges.App attribute), 32
project (sevenbridges.File attribute), 34
project (sevenbridges.models.app.App attribute), 55
project (sevenbridges.models.compound.volumes.import_destination.ImportDestination attribute), 52
project (sevenbridges.models.file.File attribute), 61
project (sevenbridges.models.task.Task attribute), 68
project (sevenbridges.Task attribute), 32
project_breakdown (sevenbridges.BillingGroupBreakdown attribute), 28
project_breakdown (sevenbridges.models.billing_breakdown.BillingGroupBreakdown attribute), 56
ProjectBreakdown (class in sevenbridges.models.compound.billing.project_breakdown), 46
projects (sevenbridges.Api attribute), 27
projects (sevenbridges.api.Api attribute), 75
projects_url (sevenbridges.Endpoints attribute), 29
projects_url (sevenbridges.models.endpoints.Endpoints attribute), 57
properties (sevenbridges.Export attribute), 35
properties (sevenbridges.models.storage_export.Export attribute), 66
provider (sevenbridges.models.compound.jobs.job_instance.Instance attribute), 49
proxies (sevenbridges.config.Profile attribute), 76
put() (sevenbridges.http.client.HttpClient method), 42

Q

query() (sevenbridges.App class method), 33
query() (sevenbridges.BillingGroup class method), 28
query() (sevenbridges.Dataset class method), 39
query() (sevenbridges.Division class method), 38
query() (sevenbridges.Export class method), 35
query() (sevenbridges.File class method), 34
query() (sevenbridges.Import class method), 36
query() (sevenbridges.Invoice class method), 27
query() (sevenbridges.Marker class method), 38
query() (sevenbridges.models.app.App class method), 55
query() (sevenbridges.models.billing_group.BillingGroup class method), 56
query() (sevenbridges.models.division.Division class method), 57
query() (sevenbridges.models.file.File class method), 61
query() (sevenbridges.models.invoice.Invoice class method), 62
query() (sevenbridges.models.marker.Marker class method), 63
query() (sevenbridges.models.project.Project class method), 65
query() (sevenbridges.models.storage_export.Export class method), 66
query() (sevenbridges.models.storage_import.Import class method), 67
query() (sevenbridges.models.task.Task class method), 68
query() (sevenbridges.models.team.Team class method), 69
query() (sevenbridges.models.volume.Volume class method), 72
query() (sevenbridges.Project class method), 30
query() (sevenbridges.Task class method), 32
query() (sevenbridges.Volume class method), 37
queued (sevenbridges.models.compound.tasks.execution_status.ExecutionS attribute), 52
QUEUED (sevenbridges.models.enums.TaskStatus attribute), 59
QUEUED (sevenbridges.TaskStatus attribute), 40
queued_duration (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 52

R

Rate (class in seven-

bridges.models.compound.limits.rate), 50
rate (sevenbridges.models.rate_limit.RateLimit attribute), 65
rate_limit (sevenbridges.Api attribute), 27
rate_limit (sevenbridges.api.Api attribute), 75
rate_limit_sleeper() (in module sevenbridges.http.error_handlers), 42
rate_limit_url (sevenbridges.Endpoints attribute), 29
rate_limit_url (sevenbridges.models.endpoints.Endpoints attribute), 57
RateLimit (class in sevenbridges.models.rate_limit), 65
raw (sevenbridges.App attribute), 33
raw (sevenbridges.models.app.App attribute), 56
READ_ONLY (sevenbridges.models.enums.VolumeAccessMode attribute), 59
READ_ONLY (sevenbridges.VolumeAccessMode attribute), 40
READ_WRITE (sevenbridges.models.enums.VolumeAccessMode attribute), 59
READ_WRITE (sevenbridges.VolumeAccessMode attribute), 40
ReadOnlyPropertyError, 41, 77
rel (sevenbridges.models.link.Link attribute), 62
reload() (sevenbridges.Division method), 39
reload() (sevenbridges.File method), 34
reload() (sevenbridges.Marker method), 38
reload() (sevenbridges.meta.resource.Resource method), 45
reload() (sevenbridges.models.division.Division method), 57
reload() (sevenbridges.models.file.File method), 61
reload() (sevenbridges.models.marker.Marker method), 63
reload() (sevenbridges.models.team.Team method), 69
remaining (sevenbridges.http.client.HttpClient attribute), 42
remaining (sevenbridges.models.compound.limits.rate.Rate attribute), 50
remove_error_handler() (sevenbridges.http.client.HttpClient method), 42
remove_member() (sevenbridges.Dataset method), 39
remove_member() (sevenbridges.models.project.Project method), 65
remove_member() (sevenbridges.models.team.Team method), 69
remove_member() (sevenbridges.models.volume.Volume method), 72
remove_member() (sevenbridges.Project method), 31
remove_member() (sevenbridges.Volume method), 38
request_id (sevenbridges.http.client.HttpClient attribute), 42
RequestTimeout, 41, 77
reset (sevenbridges.models.compound.limits.rate.Rate attribute), 50
reset_time (sevenbridges.http.client.HttpClient attribute), 42
Resource (class in sevenbridges.meta.resource), 45
resource (sevenbridges.BulkRecord attribute), 39
resource (sevenbridges.meta.collection.Collection attribute), 43
resource (sevenbridges.models.file.FileBulkRecord attribute), 61
resource (sevenbridges.models.storage_export.ExportBulkRecord attribute), 66
resource (sevenbridges.models.storage_import.ImportBulkRecord attribute), 67
ResourceMeta (class in sevenbridges.meta.resource), 45
ResourceNotModified, 41, 77
result (sevenbridges.Export attribute), 35
result (sevenbridges.Import attribute), 36
result (sevenbridges.models.storage_export.Export attribute), 66
result (sevenbridges.models.storage_import.Import attribute), 67
result() (sevenbridges.transfer.upload.Upload method), 73
resume() (sevenbridges.transfer.download.Download method), 72
resume() (sevenbridges.transfer.upload.Upload method), 73
retried (sevenbridges.models.compound.jobs.job.Job attribute), 48
retry() (in module sevenbridges.decorators), 76
retry_on_excs() (in module sevenbridges.decorators), 76
revision (sevenbridges.App attribute), 33
revision (sevenbridges.models.app.App attribute), 56
role (sevenbridges.models.team_member.TeamMember attribute), 70
run() (sevenbridges.models.task.Task method), 69
run() (sevenbridges.Task method), 32
run() (sevenbridges.transfer.download.Download method), 72
run() (sevenbridges.transfer.upload.Upload method), 73
runner_username (sevenbridges.models.compound.billing.task_breakdown.TaskBreakdow attribute), 47
RUNNING (sevenbridges.ImportExportState attribute), 40
running (sevenbridges.models.compound.tasks.execution_status.ExecutionS attribute), 52
RUNNING (sevenbridges.models.enums.ImportExportState attribute), 58
RUNNING (sevenbridges.models.enums.TaskStatus attribute), 59
RUNNING (sevenbridges.models.enums.TransferState attribute), 59
RUNNING (sevenbridges.TaskStatus attribute), 40
RUNNING (sevenbridges.TransferState attribute), 40

running_duration (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 52

S

S3 (sevenbridges.models.enums.VolumeType attribute), 59
S3 (sevenbridges.VolumeType attribute), 40
save() (sevenbridges.Dataset method), 39
save() (sevenbridges.File method), 34
save() (sevenbridges.Marker method), 38
save() (sevenbridges.Member method), 33
save() (sevenbridges.models.file.File method), 61
save() (sevenbridges.models.marker.Marker method), 63
save() (sevenbridges.models.member.Member method), 63
save() (sevenbridges.models.project.Project method), 65
save() (sevenbridges.models.task.Task method), 69
save() (sevenbridges.models.team.Team method), 69
save() (sevenbridges.models.volume.Volume method), 72
save() (sevenbridges.Project method), 31
save() (sevenbridges.Task method), 32
save() (sevenbridges.Volume method), 38
SbgError, 40, 77
send_feedback() (sevenbridges.models.actions.Actions class method), 55
ServerError, 41, 77
service (sevenbridges.models.volume.Volume attribute), 72
service (sevenbridges.Volume attribute), 38
ServiceUnavailable, 41, 77
session (sevenbridges.http.client.HttpClient attribute), 42
Settings (class in sevenbridges.models.compound.projects.settings), 50
settings (sevenbridges.models.project.Project attribute), 65
settings (sevenbridges.Project attribute), 31
sevenbridges (module), 26
sevenbridges.api (module), 74
sevenbridges.config (module), 75
sevenbridges.decorators (module), 76
sevenbridges.errors (module), 76
sevenbridges.http (module), 42
sevenbridges.http.client (module), 42
sevenbridges.http.error_handlers (module), 42
sevenbridges.meta (module), 43
sevenbridges.meta.collection (module), 43
sevenbridges.meta.comp_mutable_dict (module), 43
sevenbridges.meta.data (module), 43
sevenbridges.meta.fields (module), 44
sevenbridges.meta.resource (module), 45
sevenbridges.meta.transformer (module), 45
sevenbridges.models (module), 46
sevenbridges.models.actions (module), 55
sevenbridges.models.app (module), 55
sevenbridges.models.billing_breakdown (module), 56
sevenbridges.models.billing_group (module), 56
sevenbridges.models.compound (module), 46
sevenbridges.models.compound.billing (module), 46
sevenbridges.models.compound.billing.invoice_period (module), 46
sevenbridges.models.compound.billing.project_breakdown (module), 46
sevenbridges.models.compound.billing.task_breakdown (module), 47
sevenbridges.models.compound.error (module), 54
sevenbridges.models.compound.files (module), 47
sevenbridges.models.compound.files.download_info (module), 47
sevenbridges.models.compound.files.file_origin (module), 47
sevenbridges.models.compound.files.file_storage (module), 48
sevenbridges.models.compound.files.metadata (module), 48
sevenbridges.models.compound.jobs (module), 48
sevenbridges.models.compound.jobs.job (module), 48
sevenbridges.models.compound.jobs.job_docker (module), 49
sevenbridges.models.compound.jobs.job_instance (module), 49
sevenbridges.models.compound.jobs.job_instance_disk (module), 49
sevenbridges.models.compound.jobs.job_log (module), 49
sevenbridges.models.compound.limits (module), 50
sevenbridges.models.compound.limits.rate (module), 50
sevenbridges.models.compound.markers (module), 50
sevenbridges.models.compound.markers.position (module), 50
sevenbridges.models.compound.price (module), 54
sevenbridges.models.compound.price_breakdown (module), 54
sevenbridges.models.compound.projects (module), 50
sevenbridges.models.compound.projects.permissions (module), 50
sevenbridges.models.compound.projects.settings (module), 50
sevenbridges.models.compound.tasks (module), 51
sevenbridges.models.compound.tasks.batch_by (module), 51
sevenbridges.models.compound.tasks.batch_group (module), 51
sevenbridges.models.compound.tasks.execution_status (module), 51
sevenbridges.models.compound.tasks.input (module), 52
sevenbridges.models.compound.tasks.output (module),

52
sevenbridges.models.compound.volumes (module), 52
sevenbridges.models.compound.volumes.import_destination (module), 52
sevenbridges.models.compound.volumes.properties (module), 53
sevenbridges.models.compound.volumes.service (module), 53
sevenbridges.models.compound.volumes.volume_file (module), 53
sevenbridges.models.compound.volumes.volume_object (module), 53
sevenbridges.models.compound.volumes.volume_prefix (module), 53
sevenbridges.models.division (module), 57
sevenbridges.models.endpoints (module), 57
sevenbridges.models.enums (module), 58
sevenbridges.models.execution_details (module), 59
sevenbridges.models.file (module), 60
sevenbridges.models.invoice (module), 62
sevenbridges.models.link (module), 62
sevenbridges.models.marker (module), 62
sevenbridges.models.member (module), 63
sevenbridges.models.project (module), 63
sevenbridges.models.rate_limit (module), 65
sevenbridges.models.storage_export (module), 66
sevenbridges.models.storage_import (module), 67
sevenbridges.models.task (module), 67
sevenbridges.models.team (module), 69
sevenbridges.models.team_member (module), 70
sevenbridges.models.user (module), 70
sevenbridges.models.volume (module), 71
sevenbridges.transfer (module), 72
sevenbridges.transfer.download (module), 72
sevenbridges.transfer.upload (module), 73
sevenbridges.transfer.utils (module), 74
simple_progress_bar() (in module sevenbridges.transfer.utils), 74
size (sevenbridges.File attribute), 34
size (sevenbridges.models.compound.jobs.job_instance_disk.Disk attribute), 49
size (sevenbridges.models.file.File attribute), 61
size (sevenbridges.transfer.utils.Part attribute), 74
source (sevenbridges.Export attribute), 35
source (sevenbridges.Import attribute), 36
source (sevenbridges.models.storage_export.Export attribute), 66
source (sevenbridges.models.storage_import.Import attribute), 67
start (sevenbridges.transfer.utils.Part attribute), 74
start() (sevenbridges.transfer.download.Download method), 73
start() (sevenbridges.transfer.upload.Upload method), 73
start_time (sevenbridges.models.compound.jobs.job.Job attribute), 48
start_time (sevenbridges.models.execution_details.ExecutionDetails attribute), 59
start_time (sevenbridges.models.task.Task attribute), 69
start_time (sevenbridges.Task attribute), 32
start_time (sevenbridges.transfer.download.Download attribute), 73
start_time (sevenbridges.transfer.upload.Upload attribute), 74
started_on (sevenbridges.Export attribute), 36
started_on (sevenbridges.Import attribute), 36
started_on (sevenbridges.models.storage_export.Export attribute), 66
started_on (sevenbridges.models.storage_import.Import attribute), 67
state (sevenbridges.Export attribute), 36
state (sevenbridges.Import attribute), 36
state (sevenbridges.models.storage_export.Export attribute), 66
state (sevenbridges.models.storage_import.Import attribute), 67
state (sevenbridges.models.user.User attribute), 70
state (sevenbridges.User attribute), 28
status (sevenbridges.models.compound.error.Error attribute), 54
status (sevenbridges.models.compound.jobs.job.Job attribute), 48
status (sevenbridges.models.execution_details.ExecutionDetails attribute), 59
status (sevenbridges.models.task.Task attribute), 69
status (sevenbridges.Task attribute), 32
status (sevenbridges.transfer.download.Download attribute), 73
status (sevenbridges.transfer.upload.Upload attribute), 74
steps_completed (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 52
steps_total (sevenbridges.models.compound.tasks.execution_status.Executio attribute), 52
stop() (sevenbridges.transfer.download.Download method), 73
stop() (sevenbridges.transfer.upload.Upload method), 74
STOPPED (sevenbridges.models.enums.TransferState attribute), 59
STOPPED (sevenbridges.TransferState attribute), 40
storage (sevenbridges.File attribute), 34
storage (sevenbridges.models.compound.price_breakdown.Breakdown attribute), 54
storage (sevenbridges.models.file.File attribute), 61
storage_costs (sevenbridges.Invoice attribute), 27
storage_costs (sevenbridges.models.invoice.Invoice attribute), 62
stream() (sevenbridges.File method), 34

stream() (sevenbridges.models.file.File method), 61
StringField (class in sevenbridges.meta.fields), 44
submit() (sevenbridges.transfer.download.DPartedFile method), 72
submit() (sevenbridges.transfer.upload.UPartedFile method), 73
submit_export() (sevenbridges.Export class method), 36
submit_export() (sevenbridges.models.storage_export.Export class method), 66
submit_import() (sevenbridges.Import class method), 36
submit_import() (sevenbridges.models.storage_import.Import class method), 67
sync() (sevenbridges.App method), 33
sync() (sevenbridges.models.app.App method), 56
system_limit (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 52

T

tags (sevenbridges.File attribute), 35
tags (sevenbridges.models.file.File attribute), 61
tags (sevenbridges.models.project.Project attribute), 65
tags (sevenbridges.Project attribute), 31
Task (class in sevenbridges), 31
Task (class in sevenbridges.models.task), 67
task (sevenbridges.models.compound.files.file_origin.FileOrigin attribute), 47
task_breakdown (sevenbridges.models.compound.billing.project_breakdown.ProjectBreakdown attribute), 46
task_cost (sevenbridges.models.compound.billing.task_breakdown.TaskBreakdown attribute), 47
TaskBreakdown (class in sevenbridges.models.compound.billing.task_breakdown), 47
tasks (sevenbridges.Api attribute), 27
tasks (sevenbridges.api.Api attribute), 75
tasks_url (sevenbridges.Endpoints attribute), 29
tasks_url (sevenbridges.models.endpoints.Endpoints attribute), 57
TaskStatus (class in sevenbridges), 40
TaskStatus (class in sevenbridges.models.enums), 58
TaskValidationError, 41, 77
TB (sevenbridges.models.enums.PartSize attribute), 58
Team (class in sevenbridges.models.team), 69
TeamMember (class in sevenbridges.models.team_member), 70
teams (sevenbridges.Api attribute), 27
teams (sevenbridges.api.Api attribute), 75
THOUGHT (sevenbridges.models.enums.FeedbackType attribute), 58
time_finished (sevenbridges.models.compound.billing.task_breakdown.TaskBreakdown attribute), 47
time_started (sevenbridges.models.compound.billing.task_breakdown.TaskI attribute), 47
to (sevenbridges.models.compound.billing.invoice_period.InvoicePeriod attribute), 46
to_app() (sevenbridges.meta.transformer.Transform static method), 45
to_billing_group() (sevenbridges.meta.transformer.Transform static method), 45
to_dataset() (sevenbridges.meta.transformer.Transform static method), 45
to_datestring() (sevenbridges.meta.transformer.Transform static method), 45
to_division() (sevenbridges.meta.transformer.Transform static method), 45
to_export() (sevenbridges.meta.transformer.Transform static method), 45
to_file() (sevenbridges.meta.transformer.Transform static method), 45
to_import() (sevenbridges.meta.transformer.Transform static method), 45
to_location() (sevenbridges.meta.transformer.Transform static method), 45
to_marker() (sevenbridges.meta.transformer.Transform static method), 45
to_member() (sevenbridges.meta.transformer.Transform static method), 45
to_project() (sevenbridges.meta.transformer.Transform static method), 46
to_task() (sevenbridges.meta.transformer.Transform static method), 46
to_team() (sevenbridges.meta.transformer.Transform static method), 46
to_user() (sevenbridges.meta.transformer.Transform static method), 46
to_volume() (sevenbridges.meta.transformer.Transform static method), 46
TooManyRequests, 41, 77
total (sevenbridges.Invoice attribute), 27
total (sevenbridges.meta.collection.Collection attribute), 43
total (sevenbridges.meta.collection.VolumeCollection attribute), 43
total (sevenbridges.models.invoice.Invoice attribute), 62
total_parts() (in module sevenbridges.transfer.utils), 74
total_spending (sevenbridges.BillingGroupBreakdown attribute), 28
total_spending (sevenbridges.models.billing_breakdown.BillingGroupBreak attribute), 56
TransferState (class in sevenbridges), 40
TransferState (class in sevenbridges.models.enums), 59
Transform (class in sevenbridges.meta.transformer), 45
type (sevenbridges.BillingGroup attribute), 28
type (sevenbridges.Member attribute), 33

type (sevenbridges.models.billing_group.BillingGroup attribute), 56
type (sevenbridges.models.compound.files.file_storage.FileStorage attribute), 48
type (sevenbridges.models.compound.jobs.job_instance.Instance attribute), 49
type (sevenbridges.models.compound.jobs.job_instance_disk.Disk attribute), 49
type (sevenbridges.models.compound.volumes.volume_object.VolumeObject attribute), 53
type (sevenbridges.models.member.Member attribute), 63
type (sevenbridges.models.project.Project attribute), 65
type (sevenbridges.models.task.Task attribute), 69
type (sevenbridges.Project attribute), 31
type (sevenbridges.Task attribute), 32

U

Unauthorized, 41, 77
unit (sevenbridges.models.compound.jobs.job_instance_disk.Disk attribute), 49
UPartedFile (class in sevenbridges.transfer.upload), 73
update() (sevenbridges.meta.comp_mutable_dict.CompoundMutableDict method), 43
update() (sevenbridges.models.compound.tasks.batch_by.BatchBy method), 51
Upload (class in sevenbridges.transfer.upload), 73
upload() (sevenbridges.File class method), 35
upload() (sevenbridges.models.file.File class method), 61
UPLOAD_MINIMUM_PART_SIZE (sevenbridges.models.enums.PartSize attribute), 58
upload_url (sevenbridges.Endpoints attribute), 29
upload_url (sevenbridges.models.endpoints.Endpoints attribute), 57
url (sevenbridges.models.compound.files.download_info.DownloadInfo attribute), 47
use_interruptible_instances (sevenbridges.models.task.Task attribute), 69
use_interruptible_instances (sevenbridges.Task attribute), 32
User (class in sevenbridges), 28
User (class in sevenbridges.models.user), 70
user_url (sevenbridges.Endpoints attribute), 29
user_url (sevenbridges.models.endpoints.Endpoints attribute), 57
username (sevenbridges.Member attribute), 33
username (sevenbridges.models.member.Member attribute), 63
username (sevenbridges.models.team_member.TeamMember attribute), 70
username (sevenbridges.models.user.User attribute), 70
username (sevenbridges.User attribute), 28
users (sevenbridges.Api attribute), 27
users (sevenbridges.api.Api attribute), 75
users_url (sevenbridges.Endpoints attribute), 29
users_url (sevenbridges.models.endpoints.Endpoints attribute), 58
UuidField (class in sevenbridges.meta.fields), 44

V

valid (sevenbridges.BulkRecord attribute), 40
validate() (sevenbridges.meta.fields.BasicListField method), 44
validate() (sevenbridges.meta.fields.BooleanField method), 44
validate() (sevenbridges.meta.fields.Field method), 44
validate() (sevenbridges.meta.fields.FloatField method), 44
validate() (sevenbridges.meta.fields.IntegerField method), 44
validate() (sevenbridges.meta.fields.StringField method), 44
validate() (sevenbridges.meta.fields.UuidField method), 44
ValidationError, 41, 77
value (sevenbridges.http.client.AAHeader attribute), 42
value (sevenbridges.models.compound.tasks.batch_group.BatchGroup attribute), 51
Volume (class in sevenbridges), 36
Volume (class in sevenbridges.models.volume), 71
VOLUME (sevenbridges.FileStorageType attribute), 40
volume (sevenbridges.models.compound.files.file_storage.FileStorage attribute), 48
volume (sevenbridges.models.compound.volumes.volume_file.VolumeFile attribute), 53
volume (sevenbridges.models.compound.volumes.volume_object.VolumeO attribute), 53
volume (sevenbridges.models.compound.volumes.volume_prefix.VolumePr attribute), 54
VOLUME (sevenbridges.models.enums.FileStorageType attribute), 58
VolumeAccessMode (class in sevenbridges), 40
VolumeAccessMode (class in sevenbridges.models.enums), 59
VolumeCollection (class in sevenbridges.meta.collection), 43
VolumeFile (class in sevenbridges.models.compound.volumes.volume_file), 53
VolumeLink (class in sevenbridges.models.link), 62
VolumeObject (class in sevenbridges.models.compound.volumes.volume_object), 53
VolumePrefix (class in sevenbridges.models.compound.volumes.volume_prefix), 53
VolumeProperties (class in sevenbridges.models.compound.volumes.properties),

53
volumes (sevenbridges.Api attribute), 27
volumes (sevenbridges.api.Api attribute), 75
VolumeService (class in sevenbridges.models.compound.volumes.service), 53
VolumeType (class in sevenbridges), 40
VolumeType (class in sevenbridges.models.enums), 59

W

wait() (sevenbridges.transfer.download.Download method), 73
wait() (sevenbridges.transfer.upload.Upload method), 74
warnings (sevenbridges.models.task.Task attribute), 69
warnings (sevenbridges.Task attribute), 32

Y

YAML (sevenbridges.models.enums.AppRawFormat attribute), 58

Z

zip_code (sevenbridges.models.user.User attribute), 70
zip_code (sevenbridges.User attribute), 28