
sbg Documentation

Release 0.8.0

Seven Bridges Genomics

Jul 07, 2017

Contents

1	Content	3
1.1	Installation	3
1.2	Get the Code	3
1.3	Quickstart	3
1.4	sevenbridges package	21
	Python Module Index	61

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.

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).

You can find your authentication token on the respective pages:

- <https://igor.sbgenomics.com/developers> for the Seven Bridges Platform
- <https://cgc.sbgenomics.com/developers> for the CGC

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.

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.

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>'
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) 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>
```

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 fails it will raise an exception

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.

```
# 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. Remove a member from the project - `project.remove_member()`
4. List files from the project - `project.get_files()`
5. Add files to the project - `project.add_files()` - you can add a single File or a Collection of files
6. List apps from the project - `project.get_apps()`
7. List tasks from the project - `project.get_tasks()`

Manage billing

There are several methods on the `Api` object to 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 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()`

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()
```

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](#).

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.

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
task_name = 'my-first-task'

# Project in which I want to run a task.
project_id = '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=task_name, project=project_id, 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
task_name = 'my-first-task'

# Project in which to run the task.
project_id = '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='my-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=task_name, project=project_id, 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.')
```


sevenbridges package

sevenbridges-python

copyright 2016 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*)

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`

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`

users

alias of `User`

volumes

alias of `Volume`

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

Bases: `object`

Utility configuration class.

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

  Central resource for managing invoices.

  analysis_costs

  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.

  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.

  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
    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
    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_email (*email, permissions=None*)

Add a member to the project using member email. :param email: Member email. :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 de-
scription: Project description. :param tags: Project tags. :param settings: Project settings. :param api: Api
instance. :return:

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, 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 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
  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 SbgError 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

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.

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*)

Installs and app. :param id: App identifier. :param raw: Raw cwl data. :param api: Api instance. :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

```

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

    Central resource for managing project members.

    email = None

    equals (other)

    href = None

    permissions

    save (obj, *args, **kwargs)

    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.

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

    created_on = None

    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, names=None, metadata=None, origin=None, tags=None, offset=None, limit=None, api=None)
        Query ( List ) projects :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 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.

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 state.
: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.

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

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

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.

href = None

id = None

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.

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

service

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.HttpClient` (*url=None, token=None, oauth_token=None, config=None, timeout=None, proxies=None, error_handlers=None*)

Bases: `object`

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

add_error_handler (*handler*)

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

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

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*)

Utility method to fetch config vars using ini section profile :param profiles: profile name. :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*)

Pauses the execution if response status code is > 500. :param api: Api instance. :param response: requests.Response object

`sevenbridges.http.error_handlers.maintenance_sleeper` (*api, response*)

Pauses the execution if sevenbridges api is under maintenance. :param api: Api instance. :param response: requests.Response object.

`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**

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, valida-
                                                tor=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*)

static to_billing_group (*billing_group*)

static to_datestring (*d*)

static to_file (*file_*)

static to_project (*project*)

static to_task (*task*)

static to_user (*user*)

static to_volume (*volume*)

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.

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

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.

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.

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.

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.

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
    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
    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.
    disk
    equals (other)
    id = None
    provider = None
    type = 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.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 sevenbridges 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.

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

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.
copy ()

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.
copy ()

sevenbridges.models.compound.volumes package

Submodules

sevenbridges.models.compound.volumes.credentials module

class sevenbridges.models.compound.volumes.credentials.**VolumeCredentials** (**kwargs)
Bases: *sevenbridges.meta.comp_mutable_dict.CompoundMutableDict, sevenbridges.meta.resource.Resource*
Volume permissions resource.

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.
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.

equals (*other*)

location = 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

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

equals (*other*)

Submodules

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.

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*)
 Installs and app. :param id: App identifier. :param raw: Raw cwl data. :param api: Api instance. :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

sevenbridges.models.billing_breakdown module

class `sevenbridges.models.billing_breakdown.BillingGroupBreakdown` (**kwargs)
 Bases: `sevenbridges.meta.resource.Resource`
 Central resource for managing billing group breakdowns.

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.
    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.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
    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.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.

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.

copy (*project, name=None*)

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

created_on = None

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*, *names=None*, *metadata=None*, *origin=None*, *tags=None*, *offset=None*, *limit=None*, *api=None*)

Query (List) projects :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 *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.

sevenbridges.models.invoice module

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

Bases: `sevenbridges.meta.resource.Resource`

Central resource for managing invoices.

analysis_costs

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.

equals (*other*)

href = None

method = None

rel = None

sevenbridges.models.member module

class `sevenbridges.models.member.Member` (**kwargs)

Bases: `sevenbridges.meta.resource.Resource`

Central resource for managing project members.

email = None

equals (*other*)

href = None

permissions

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

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

Add a member to the project using member email. :param email: Member email. :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:

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.storage_export module

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

Bases: *sevenbridges.meta.resource.Resource*

Central resource for managing exports.

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.

sevenbridges.models.storage_import module

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

Bases: `sevenbridges.meta.resource.Resource`

Central resource for managing imports.

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.

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, 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

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

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 SbgError 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

warnings = 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

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

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*, *de-*
scription=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

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.

href = None

id = None

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.

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*)

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

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

users
alias of User

volumes
alias of Volume

sevenbridges.config module

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

Bases: object

Utility configuration class.

class `sevenbridges.config.Profile` (*profile*)

Bases: object

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

PROXIES = `'/home/docs/.sevenbridges/sevenbridges-python/config'`

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_exc`s (*exc*s, *retry_count*=3, *delay*=1)

Retry decorator used to retry callables on for specific exceptions.

Parameters

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

Returns Wrapped function object.

sevenbridges.errors module

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, 21
sevenbridges.api, 56
sevenbridges.config, 57
sevenbridges.decorators, 58
sevenbridges.errors, 58
sevenbridges.http, 32
sevenbridges.http.client, 32
sevenbridges.http.error_handlers, 33
sevenbridges.meta, 33
sevenbridges.meta.collection, 33
sevenbridges.meta.comp_mutable_dict, 33
sevenbridges.meta.data, 34
sevenbridges.meta.fields, 34
sevenbridges.meta.resource, 35
sevenbridges.meta.transformer, 35
sevenbridges.models, 36
sevenbridges.models.app, 43
sevenbridges.models.billing_breakdown, 43
sevenbridges.models.billing_group, 44
sevenbridges.models.compound, 36
sevenbridges.models.compound.billing, 36
sevenbridges.models.compound.billing.invoice_period, 36
sevenbridges.models.compound.billing.project_breakdown, 36
sevenbridges.models.compound.billing.task_breakdown, 36
sevenbridges.models.compound.error, 42
sevenbridges.models.compound.files, 37
sevenbridges.models.compound.files.download_info, 37
sevenbridges.models.compound.files.file_origin, 37
sevenbridges.models.compound.files.file_storage, 37
sevenbridges.models.compound.files.metadata, 37
sevenbridges.models.compound.jobs, 38
sevenbridges.models.compound.jobs.job, 38
sevenbridges.models.compound.jobs.job_docker, 38
sevenbridges.models.compound.jobs.job_instance, 38
sevenbridges.models.compound.jobs.job_log, 39
sevenbridges.models.compound.price, 42
sevenbridges.models.compound.projects, 39
sevenbridges.models.compound.projects.permissions, 39
sevenbridges.models.compound.projects.settings, 39
sevenbridges.models.compound.tasks, 39
sevenbridges.models.compound.tasks.batch_by, 39
sevenbridges.models.compound.tasks.batch_group, 40
sevenbridges.models.compound.tasks.execution_status, 40
sevenbridges.models.compound.tasks.input, 41
sevenbridges.models.compound.tasks.output, 41
sevenbridges.models.compound.volumes, 41
sevenbridges.models.compound.volumes.credentials, 41
sevenbridges.models.compound.volumes.import_destination, 41
sevenbridges.models.compound.volumes.properties, 41
sevenbridges.models.compound.volumes.service, 42
sevenbridges.models.compound.volumes.volume_file, 42

- sevenbridges.models.endpoints, 44
- sevenbridges.models.enums, 45
- sevenbridges.models.execution_details,
46
- sevenbridges.models.file, 46
- sevenbridges.models.invoice, 48
- sevenbridges.models.link, 48
- sevenbridges.models.member, 48
- sevenbridges.models.project, 49
- sevenbridges.models.storage_export, 50
- sevenbridges.models.storage_import, 51
- sevenbridges.models.task, 51
- sevenbridges.models.user, 53
- sevenbridges.models.volume, 53
- sevenbridges.transfer, 54
- sevenbridges.transfer.download, 54
- sevenbridges.transfer.upload, 55
- sevenbridges.transfer.utils, 56

A

- abort() (sevenbridges.models.task.Task method), 51
- abort() (sevenbridges.Task method), 25
- aborted (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
- ABORTED (sevenbridges.models.enums.TaskStatus attribute), 45
- ABORTED (sevenbridges.models.enums.TransferState attribute), 45
- ABORTED (sevenbridges.TaskStatus attribute), 31
- ABORTED (sevenbridges.TransferState attribute), 30
- access_mode (sevenbridges.models.volume.Volume attribute), 53
- access_mode (sevenbridges.Volume attribute), 29
- account_limit (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
- action_url (sevenbridges.Endpoints attribute), 23
- action_url (sevenbridges.models.endpoints.Endpoints attribute), 44
- actions (sevenbridges.Api attribute), 21
- actions (sevenbridges.api.Api attribute), 56
- active (sevenbridges.models.volume.Volume attribute), 53
- active (sevenbridges.Volume attribute), 29
- add_callback() (sevenbridges.transfer.download.Download method), 54
- add_callback() (sevenbridges.transfer.upload.Upload method), 55
- add_error_handler() (sevenbridges.http.client.HttpClient method), 32
- add_files() (sevenbridges.models.project.Project method), 49
- add_files() (sevenbridges.Project method), 23
- add_member() (sevenbridges.models.project.Project method), 49
- add_member() (sevenbridges.Project method), 24
- add_member_email() (sevenbridges.models.project.Project method), 49
- add_member_email() (sevenbridges.Project method), 24
- add_progress_callback() (sevenbridges.transfer.download.Download method), 54
- add_progress_callback() (sevenbridges.transfer.upload.Upload method), 55
- address (sevenbridges.models.user.User attribute), 53
- address (sevenbridges.User attribute), 23
- affiliation (sevenbridges.models.user.User attribute), 53
- affiliation (sevenbridges.User attribute), 23
- all() (sevenbridges.meta.collection.Collection method), 33
- amount (sevenbridges.models.compound.price.Price attribute), 42
- analysis_costs (sevenbridges.Invoice attribute), 22
- analysis_costs (sevenbridges.models.invoice.Invoice attribute), 48
- analysis_spending (sevenbridges.models.compound.billing.project_breakdown.ProjectBreakdown attribute), 36
- Api (class in sevenbridges), 21
- Api (class in sevenbridges.api), 56
- api_endpoint (sevenbridges.config.Profile attribute), 57
- App (class in sevenbridges), 26
- App (class in sevenbridges.models.app), 43
- app (sevenbridges.models.task.Task attribute), 51
- app (sevenbridges.Task attribute), 25
- apps (sevenbridges.Api attribute), 21
- apps (sevenbridges.api.Api attribute), 57
- apps_url (sevenbridges.Endpoints attribute), 23
- apps_url (sevenbridges.models.endpoints.Endpoints attribute), 44
- auth_token (sevenbridges.config.Profile attribute), 57

B

- BadRequest, 31, 58
- balance (sevenbridges.BillingGroup attribute), 22
- balance (sevenbridges.models.billing_group.BillingGroup attribute), 44

bandwidth (sevenbridges.transfer.utils.Progress attribute), 56

BasicListField (class in sevenbridges.meta.fields), 34

batch (sevenbridges.models.task.Task attribute), 51

batch (sevenbridges.Task attribute), 25

batch_by (sevenbridges.models.task.Task attribute), 51

batch_by (sevenbridges.Task attribute), 25

batch_group (sevenbridges.models.task.Task attribute), 51

batch_group (sevenbridges.Task attribute), 25

batch_input (sevenbridges.models.task.Task attribute), 51

batch_input (sevenbridges.Task attribute), 25

BatchBy (class in sevenbridges.models.compound.tasks.batch_by), 39

BatchGroup (class in sevenbridges.models.compound.tasks.batch_group), 40

billing_group (sevenbridges.models.project.Project attribute), 49

billing_group (sevenbridges.Project attribute), 24

billing_groups (sevenbridges.Api attribute), 21

billing_groups (sevenbridges.api.Api attribute), 57

billing_url (sevenbridges.Endpoints attribute), 23

billing_url (sevenbridges.models.endpoints.Endpoints attribute), 44

BillingGroup (class in sevenbridges), 22

BillingGroup (class in sevenbridges.models.billing_group), 44

BillingGroupBreakdown (class in sevenbridges), 22

BillingGroupBreakdown (class in sevenbridges.models.billing_breakdown), 43

BooleanField (class in sevenbridges.meta.fields), 34

breakdown (sevenbridges.models.compound.price.Price attribute), 42

breakdown() (sevenbridges.BillingGroup method), 22

breakdown() (sevenbridges.models.billing_group.BillingGroup method), 44

bytes_done (sevenbridges.transfer.utils.Progress attribute), 56

C

check_for_error() (in module sevenbridges.decorators), 58

checksum (sevenbridges.models.compound.jobs.job_docker.job_docker attribute), 38

city (sevenbridges.models.user.User attribute), 53

city (sevenbridges.User attribute), 23

code (sevenbridges.models.compound.error.Error attribute), 42

Collection (class in sevenbridges.meta.collection), 33

command_line (sevenbridges.models.compound.jobs.job_docker.job_docker attribute), 38

COMPLETED (sevenbridges.ImportExportState attribute), 31

completed (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40

COMPLETED (sevenbridges.models.enums.ImportExportState attribute), 45

COMPLETED (sevenbridges.models.enums.TaskStatus attribute), 45

COMPLETED (sevenbridges.models.enums.TransferState attribute), 46

COMPLETED (sevenbridges.TaskStatus attribute), 31

COMPLETED (sevenbridges.TransferState attribute), 30

CompoundField (class in sevenbridges.meta.fields), 34

CompoundListField (class in sevenbridges.meta.fields), 34

CompoundMutableDict (class in sevenbridges.meta.comp_mutable_dict), 33

Config (class in sevenbridges), 21

Config (class in sevenbridges.config), 57

config_vars() (in module sevenbridges.http.client), 33

Conflict, 32, 58

copy() (sevenbridges.App method), 26

copy() (sevenbridges.File method), 27

copy() (sevenbridges.models.app.App method), 43

copy() (sevenbridges.models.compound.tasks.input.Input method), 41

copy() (sevenbridges.models.compound.tasks.output.Output method), 41

copy() (sevenbridges.models.file.File method), 46

country (sevenbridges.models.user.User attribute), 53

country (sevenbridges.User attribute), 23

create() (sevenbridges.models.project.Project class method), 49

create() (sevenbridges.models.task.Task class method), 51

create() (sevenbridges.Project class method), 24

create() (sevenbridges.Task class method), 25

create_google_volume() (sevenbridges.models.volume.Volume class method), 53

create_google_volume() (sevenbridges.Volume class method), 29

create_revision() (sevenbridges.App class method), 26

create_revision() (sevenbridges.models.app.App class method), 43

create_s3_volume() (sevenbridges.models.volume.Volume class method), 53

create_s3_volume() (sevenbridges.Volume class method), 29

created_by (sevenbridges.models.task.Task attribute), 52

created_by (sevenbridges.Task attribute), 25

created_on (sevenbridges.File attribute), 27

created_on (sevenbridges.models.file.File attribute), 46

created_on (sevenbridges.models.volume.Volume attribute), 53
 created_on (sevenbridges.Volume attribute), 30
 created_time (sevenbridges.models.task.Task attribute), 52
 created_time (sevenbridges.Task attribute), 25
 CREATING (sevenbridges.models.enums.TaskStatus attribute), 45
 CREATING (sevenbridges.TaskStatus attribute), 31
 CREDENTIALS (sevenbridges.config.Profile attribute), 57
 currency (sevenbridges.models.compound.price.Price attribute), 42

D

DataContainer (class in sevenbridges.meta.data), 34
 DateTimeField (class in sevenbridges.meta.fields), 34
 delete() (sevenbridges.http.client.HttpClient method), 32
 delete() (sevenbridges.meta.resource.Resource method), 35
 description (sevenbridges.models.project.Project attribute), 49
 description (sevenbridges.models.task.Task attribute), 52
 description (sevenbridges.models.volume.Volume attribute), 54
 description (sevenbridges.Project attribute), 24
 description (sevenbridges.Task attribute), 25
 description (sevenbridges.Volume attribute), 30
 destination (sevenbridges.Export attribute), 28
 destination (sevenbridges.Import attribute), 29
 destination (sevenbridges.models.storage_export.Export attribute), 50
 destination (sevenbridges.models.storage_import.Import attribute), 51
 DictField (class in sevenbridges.meta.fields), 34
 disabled (sevenbridges.BillingGroup attribute), 22
 disabled (sevenbridges.models.billing_group.BillingGroup attribute), 44
 disk (sevenbridges.models.compound.jobs.job_instance.Instance attribute), 38
 docker (sevenbridges.models.compound.jobs.job.Job attribute), 38
 done() (sevenbridges.transfer.download.DPartedFile method), 54
 done() (sevenbridges.transfer.upload.UPartedFile method), 55
 Download (class in sevenbridges.transfer.download), 54
 download() (sevenbridges.File method), 27
 download() (sevenbridges.models.file.File method), 46
 download_info() (sevenbridges.File method), 27
 download_info() (sevenbridges.models.file.File method), 47
 DOWNLOAD_MINIMUM_PART_SIZE (sevenbridges.models.enums.PartSize attribute),

45
 DownloadInfo (class in sevenbridges.models.compound.files.download_info), 37
 DPartedFile (class in sevenbridges.transfer.download), 54
 DRAFT (sevenbridges.models.enums.TaskStatus attribute), 45
 DRAFT (sevenbridges.TaskStatus attribute), 31
 duration (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
 duration (sevenbridges.transfer.download.Download attribute), 54
 duration (sevenbridges.transfer.upload.Upload attribute), 55
 duration (sevenbridges.transfer.utils.Progress attribute), 56

E

email (sevenbridges.Member attribute), 27
 email (sevenbridges.models.member.Member attribute), 48
 email (sevenbridges.models.user.User attribute), 53
 email (sevenbridges.User attribute), 23
 end_time (sevenbridges.models.compound.jobs.job.Job attribute), 38
 end_time (sevenbridges.models.execution_details.ExecutionDetails attribute), 46
 end_time (sevenbridges.models.task.Task attribute), 52
 end_time (sevenbridges.Task attribute), 25
 Endpoints (class in sevenbridges), 23
 Endpoints (class in sevenbridges.models.endpoints), 44
 endpoints (sevenbridges.Api attribute), 21
 endpoints (sevenbridges.api.Api attribute), 57
 equals() (sevenbridges.App method), 26
 equals() (sevenbridges.BillingGroup method), 22
 equals() (sevenbridges.BillingGroupBreakdown method), 22
 equals() (sevenbridges.Endpoints method), 23
 equals() (sevenbridges.Export method), 28
 equals() (sevenbridges.File method), 27
 equals() (sevenbridges.Import method), 29
 equals() (sevenbridges.Invoice method), 22
 equals() (sevenbridges.Member method), 27
 equals() (sevenbridges.meta.comp_mutable_dict.CompoundMutableDict method), 33
 equals() (sevenbridges.models.app.App method), 43
 equals() (sevenbridges.models.billing_breakdown.BillingGroupBreakdown method), 43
 equals() (sevenbridges.models.billing_group.BillingGroup method), 44
 equals() (sevenbridges.models.compound.billing.invoice_period.InvoicePeriod method), 36
 equals() (sevenbridges.models.compound.billing.project_breakdown.ProjectBreakdown method), 36

[equals\(\) \(sevenbridges.models.compound.billing.task_breakdown.TaskBreakdown method\), 36](#)
[equals\(\) \(sevenbridges.models.compound.error.Error method\), 42](#)
[equals\(\) \(sevenbridges.models.compound.files.download_info.DownloadInfo method\), 37](#)
[equals\(\) \(sevenbridges.models.compound.files.file_origin.FileOrigin method\), 37](#)
[equals\(\) \(sevenbridges.models.compound.files.file_storage.FileStorage method\), 37](#)
[equals\(\) \(sevenbridges.models.compound.jobs.job.Job method\), 38](#)
[equals\(\) \(sevenbridges.models.compound.jobs.job_docker.JobDocker method\), 38](#)
[equals\(\) \(sevenbridges.models.compound.jobs.job_instance.Instance method\), 38](#)
[equals\(\) \(sevenbridges.models.compound.price.Price method\), 42](#)
[equals\(\) \(sevenbridges.models.compound.tasks.batch_by.BatchBy method\), 39](#)
[equals\(\) \(sevenbridges.models.compound.tasks.batch_group.BatchGroup method\), 40](#)
[equals\(\) \(sevenbridges.models.compound.tasks.execution_status.ExecutionStatus method\), 40](#)
[equals\(\) \(sevenbridges.models.compound.volumes.import_exports.ImportExports method\), 41](#)
[equals\(\) \(sevenbridges.models.compound.volumes.volume_file.VolumeFile method\), 42](#)
[equals\(\) \(sevenbridges.models.endpoints.Endpoints method\), 44](#)
[equals\(\) \(sevenbridges.models.execution_details.ExecutionDetails method\), 46](#)
[equals\(\) \(sevenbridges.models.file.File method\), 47](#)
[equals\(\) \(sevenbridges.models.invoice.Invoice method\), 48](#)
[equals\(\) \(sevenbridges.models.link.Link method\), 48](#)
[equals\(\) \(sevenbridges.models.member.Member method\), 48](#)
[equals\(\) \(sevenbridges.models.project.Project method\), 49](#)
[equals\(\) \(sevenbridges.models.storage_export.Export method\), 50](#)
[equals\(\) \(sevenbridges.models.storage_import.Import method\), 51](#)
[equals\(\) \(sevenbridges.models.task.Task method\), 52](#)
[equals\(\) \(sevenbridges.models.user.User method\), 53](#)
[equals\(\) \(sevenbridges.models.volume.Volume method\), 54](#)
[equals\(\) \(sevenbridges.Project method\), 24](#)
[equals\(\) \(sevenbridges.Task method\), 25](#)
[equals\(\) \(sevenbridges.User method\), 23](#)
[equals\(\) \(sevenbridges.Volume method\), 30](#)
[Error \(class in sevenbridges.models.compound.error\), 42](#)
[error \(sevenbridges.Export attribute\), 28](#)
[error \(sevenbridges.models.storage_export.Export attribute\), 50](#)
[error \(sevenbridges.models.storage_import.Import attribute\), 51](#)
[errors \(sevenbridges.models.task.Task attribute\), 52](#)
[FileOrigin \(sevenbridges.Task attribute\), 25](#)
[executed_by \(sevenbridges.models.task.Task attribute\), 52](#)
[executed_by \(sevenbridges.Task attribute\), 25](#)
[execution_duration \(sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute\), 40](#)
[execution_status \(sevenbridges.models.task.Task attribute\), 52](#)
[execution_status \(sevenbridges.Task attribute\), 25](#)
[ExecutionDetails \(class in sevenbridges.models.execution_details\), 46](#)
[ExecutionDetailsInvalidTaskType, 32, 58](#)
[ExecutionStatus \(class in sevenbridges.models.compound.tasks.execution_status\), 40](#)
[Exclude \(class in sevenbridges\), 28](#)
[Export \(class in sevenbridges.models.storage_export\), 50](#)
[exports \(sevenbridges.Api attribute\), 21](#)
[exports \(sevenbridges.api.Api attribute\), 57](#)

F

[FAILED \(sevenbridges.ImportExportState attribute\), 31](#)
[failed \(sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute\), 40](#)
[FAILED \(sevenbridges.models.enums.ImportExportState attribute\), 45](#)
[FAILED \(sevenbridges.models.enums.TaskStatus attribute\), 45](#)
[FAILED \(sevenbridges.models.enums.TransferState attribute\), 46](#)
[FAILED \(sevenbridges.TaskStatus attribute\), 31](#)
[FAILED \(sevenbridges.TransferState attribute\), 30](#)
[FeedbackType \(class in sevenbridges.models.enums\), 45](#)
[fetch\(\) \(sevenbridges.meta.data.DataContainer method\), 34](#)
[Field \(class in sevenbridges.meta.fields\), 34](#)
[fields \(sevenbridges.models.compound.tasks.batch_group.BatchGroup attribute\), 40](#)
[File \(class in sevenbridges\), 27](#)
[File \(class in sevenbridges.models.file\), 46](#)
[file_name \(sevenbridges.transfer.upload.Upload attribute\), 55](#)
[file_size \(sevenbridges.transfer.utils.Progress attribute\), 56](#)
[FileOrigin \(class in sevenbridges.models.compound.files.file_origin\), 37](#)

files (sevenbridges.Api attribute), 21
 files (sevenbridges.api.Api attribute), 57
 files_url (sevenbridges.Endpoints attribute), 23
 files_url (sevenbridges.models.endpoints.Endpoints attribute), 44
 FileStorage (class in sevenbridges.models.compound.files.file_storage), 37
 FileStorageType (class in sevenbridges), 30
 FileStorageType (class in sevenbridges.models.enums), 45
 finished_on (sevenbridges.Export attribute), 28
 finished_on (sevenbridges.Import attribute), 29
 finished_on (sevenbridges.models.storage_export.Export attribute), 50
 finished_on (sevenbridges.models.storage_import.Import attribute), 51
 first_name (sevenbridges.models.user.User attribute), 53
 first_name (sevenbridges.User attribute), 23
 FloatField (class in sevenbridges.meta.fields), 34
 Forbidden, 31, 58
 format_proxies() (in module sevenbridges.config), 57
 from_ (sevenbridges.models.compound.billing.invoice_period.GOOGLEResource attribute), 36

G

GB (sevenbridges.models.enums.PartSize attribute), 45
 general_error_sleeper() (in module sevenbridges.http.error_handlers), 33
 generate_session() (in module sevenbridges.http.client), 33
 get() (sevenbridges.Endpoints class method), 23
 get() (sevenbridges.http.client.HttpClient method), 32
 get() (sevenbridges.meta.resource.Resource class method), 35
 get() (sevenbridges.models.endpoints.Endpoints class method), 44
 get() (sevenbridges.models.user.User class method), 53
 get() (sevenbridges.User class method), 23
 get_apps() (sevenbridges.models.project.Project method), 49
 get_apps() (sevenbridges.Project method), 24
 get_batch_children() (sevenbridges.models.task.Task method), 52
 get_batch_children() (sevenbridges.Task method), 25
 get_execution_details() (sevenbridges.models.task.Task method), 52
 get_execution_details() (sevenbridges.Task method), 25
 get_exports() (sevenbridges.models.project.Project method), 49
 get_exports() (sevenbridges.models.volume.Volume method), 54
 get_exports() (sevenbridges.Project method), 24
 get_exports() (sevenbridges.Volume method), 30

get_files() (sevenbridges.models.project.Project method), 49
 get_files() (sevenbridges.Project method), 24
 get_imports() (sevenbridges.models.project.Project method), 49
 get_imports() (sevenbridges.models.volume.Volume method), 54
 get_imports() (sevenbridges.Project method), 24
 get_imports() (sevenbridges.Volume method), 30
 get_members() (sevenbridges.models.project.Project method), 49
 get_members() (sevenbridges.Project method), 24
 get_parts() (sevenbridges.transfer.download.DPartedFile method), 54
 get_parts() (sevenbridges.transfer.upload.UPartedFile method), 55
 get_revision() (sevenbridges.App class method), 26
 get_revision() (sevenbridges.models.app.App class method), 43
 get_tasks() (sevenbridges.models.project.Project method), 49
 get_tasks() (sevenbridges.Project method), 24
 GOOGLEPeriod (sevenbridges.models.enums.VolumeType attribute), 46
 GOOGLE (sevenbridges.VolumeType attribute), 30

H

href (sevenbridges.App attribute), 26
 href (sevenbridges.BillingGroup attribute), 22
 href (sevenbridges.BillingGroupBreakdown attribute), 22
 href (sevenbridges.Export attribute), 28
 href (sevenbridges.File attribute), 27
 href (sevenbridges.Import attribute), 29
 href (sevenbridges.Invoice attribute), 22
 href (sevenbridges.Member attribute), 27
 href (sevenbridges.models.app.App attribute), 43
 href (sevenbridges.models.billing_breakdown.BillingGroupBreakdown attribute), 43
 href (sevenbridges.models.billing_group.BillingGroup attribute), 44
 href (sevenbridges.models.compound.billing.project_breakdown.ProjectBreakdown attribute), 36
 href (sevenbridges.models.compound.billing.task_breakdown.TaskBreakdown attribute), 36
 href (sevenbridges.models.execution_details.ExecutionDetails attribute), 46
 href (sevenbridges.models.file.File attribute), 47
 href (sevenbridges.models.invoice.Invoice attribute), 48
 href (sevenbridges.models.link.Link attribute), 48
 href (sevenbridges.models.member.Member attribute), 48
 href (sevenbridges.models.project.Project attribute), 49
 href (sevenbridges.models.storage_export.Export attribute), 50

- href (sevenbridges.models.storage_import.Import attribute), 51
 - href (sevenbridges.models.task.Task attribute), 52
 - href (sevenbridges.models.user.User attribute), 53
 - href (sevenbridges.models.volume.Volume attribute), 54
 - href (sevenbridges.Project attribute), 24
 - href (sevenbridges.Task attribute), 25
 - href (sevenbridges.User attribute), 23
 - href (sevenbridges.Volume attribute), 30
 - HrefField (class in sevenbridges.meta.fields), 34
 - HttpClient (class in sevenbridges.http.client), 32
- I**
- id (sevenbridges.App attribute), 26
 - id (sevenbridges.BillingGroup attribute), 22
 - id (sevenbridges.Export attribute), 28
 - id (sevenbridges.File attribute), 27
 - id (sevenbridges.Import attribute), 29
 - id (sevenbridges.Invoice attribute), 22
 - id (sevenbridges.models.app.App attribute), 43
 - id (sevenbridges.models.billing_group.BillingGroup attribute), 44
 - id (sevenbridges.models.compound.jobs.job_instance.Instance attribute), 38
 - id (sevenbridges.models.file.File attribute), 47
 - id (sevenbridges.models.invoice.Invoice attribute), 48
 - id (sevenbridges.models.project.Project attribute), 49
 - id (sevenbridges.models.storage_export.Export attribute), 50
 - id (sevenbridges.models.storage_import.Import attribute), 51
 - id (sevenbridges.models.task.Task attribute), 52
 - id (sevenbridges.models.volume.Volume attribute), 54
 - id (sevenbridges.Project attribute), 24
 - id (sevenbridges.Task attribute), 25
 - id (sevenbridges.Volume attribute), 30
 - IDEA (sevenbridges.models.enums.FeedbackType attribute), 45
 - Import (class in sevenbridges), 29
 - Import (class in sevenbridges.models.storage_import), 51
 - ImportDestination (class in sevenbridges.models.compound.volumes.import_destination), 41
 - ImportExportState (class in sevenbridges), 31
 - ImportExportState (class in sevenbridges.models.enums), 45
 - imports (sevenbridges.Api attribute), 21
 - imports (sevenbridges.api.Api attribute), 57
 - inplace_reload() (in module sevenbridges.decorators), 58
 - Input (class in sevenbridges.models.compound.tasks.input), 41
 - inputs (sevenbridges.models.task.Task attribute), 52
 - inputs (sevenbridges.Task attribute), 25
 - install_app() (sevenbridges.App class method), 26
 - install_app() (sevenbridges.models.app.App class method), 43
 - Instance (class in sevenbridges.models.compound.jobs.job_instance), 38
 - instance (sevenbridges.models.compound.jobs.job.Job attribute), 38
 - instance_init (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
 - IntegerField (class in sevenbridges.meta.fields), 34
 - Invoice (class in sevenbridges), 21
 - Invoice (class in sevenbridges.models.invoice), 48
 - invoice_period (sevenbridges.Invoice attribute), 22
 - invoice_period (sevenbridges.models.invoice.Invoice attribute), 48
 - InvoicePeriod (class in sevenbridges.models.compound.billing.invoice_period), 36
 - invoices (sevenbridges.Api attribute), 21
 - invoices (sevenbridges.api.Api attribute), 57
 - items() (sevenbridges.meta.comp_mutable_dict.CompoundMutableDict method), 34
- J**
- Job (class in sevenbridges.models.compound.jobs.job), 38
 - JobDocker (class in sevenbridges.models.compound.jobs.job_docker), 38
 - jobs (sevenbridges.models.execution_details.ExecutionDetails attribute), 46
- K**
- KB (sevenbridges.models.enums.PartSize attribute), 45
- L**
- last_name (sevenbridges.models.user.User attribute), 53
 - last_name (sevenbridges.User attribute), 23
 - limit (sevenbridges.http.client.HttpClient attribute), 32
 - Link (class in sevenbridges.models.link), 48
 - LocalFileAlreadyExists, 32, 58
 - Location (sevenbridges.models.compound.files.file_storage.FileStorage attribute), 37
 - location (sevenbridges.models.compound.volumes.volume_file.VolumeFile attribute), 42
 - Logs (class in sevenbridges.models.compound.jobs.job_log), 39
 - logs (sevenbridges.models.compound.jobs.job.Job attribute), 38
- M**
- maintenance_sleeper() (in module sevenbridges.http.error_handlers), 33

- map_input_output() (in module sevenbridges.models.compound.tasks), 39
- markers (sevenbridges.Api attribute), 21
- markers (sevenbridges.api.Api attribute), 57
- MAXIMUM_OBJECT_SIZE (sevenbridges.models.enums.PartSize attribute), 45
- MAXIMUM_TOTAL_PARTS (sevenbridges.models.enums.PartSize attribute), 45
- MAXIMUM_UPLOAD_SIZE (sevenbridges.models.enums.PartSize attribute), 45
- MB (sevenbridges.models.enums.PartSize attribute), 45
- me() (sevenbridges.models.user.User class method), 53
- me() (sevenbridges.User class method), 23
- Member (class in sevenbridges), 26
- Member (class in sevenbridges.models.member), 48
- message (sevenbridges.models.compound.error.Error attribute), 42
- message (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
- message (sevenbridges.models.execution_details.ExecutionDetails attribute), 46
- message_code (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
- Metadata (class in sevenbridges.models.compound.files.metadata), 37
- metadata (sevenbridges.File attribute), 27
- metadata (sevenbridges.models.file.File attribute), 47
- method (sevenbridges.models.link.Link attribute), 48
- MethodNotAllowed, 32, 58
- modified_on (sevenbridges.File attribute), 27
- modified_on (sevenbridges.models.file.File attribute), 47
- modified_on (sevenbridges.models.volume.Volume attribute), 54
- modified_on (sevenbridges.Volume attribute), 30
- more_info (sevenbridges.models.compound.error.Error attribute), 42
- ## N
- name (sevenbridges.App attribute), 26
- name (sevenbridges.BillingGroup attribute), 22
- name (sevenbridges.File attribute), 27
- name (sevenbridges.models.app.App attribute), 43
- name (sevenbridges.models.billing_group.BillingGroup attribute), 44
- name (sevenbridges.models.compound.jobs.job.Job attribute), 38
- name (sevenbridges.models.compound.volumes.import_destination.ImportDestination attribute), 41
- name (sevenbridges.models.file.File attribute), 47
- name (sevenbridges.models.project.Project attribute), 49
- name (sevenbridges.models.task.Task attribute), 52
- name (sevenbridges.models.volume.Volume attribute), 54
- name (sevenbridges.Project attribute), 24
- name (sevenbridges.Task attribute), 25
- name (sevenbridges.Volume attribute), 30
- next_page() (sevenbridges.meta.collection.Collection method), 33
- NotFound, 31, 58
- num_of_parts (sevenbridges.transfer.utils.Progress attribute), 56
- ## O
- ObjectIdField (class in sevenbridges.meta.fields), 34
- origin (sevenbridges.File attribute), 27
- origin (sevenbridges.models.file.File attribute), 47
- Output (class in sevenbridges.models.compound.tasks.output), 41
- outputs (sevenbridges.models.task.Task attribute), 52
- outputs (sevenbridges.Task attribute), 25
- override (sevenbridges.models.export.Export attribute), 28
- override (sevenbridges.Import attribute), 29
- override (sevenbridges.models.storage_export.Export attribute), 50
- override (sevenbridges.models.storage_import.Import attribute), 51
- owner (sevenbridges.BillingGroup attribute), 22
- owner (sevenbridges.models.billing_group.BillingGroup attribute), 44
- ## P
- PaginationError, 31, 58
- parent (sevenbridges.models.task.Task attribute), 52
- parent (sevenbridges.Task attribute), 25
- Part (class in sevenbridges.transfer.utils), 56
- parts_done (sevenbridges.transfer.utils.Progress attribute), 56
- PartSize (class in sevenbridges.models.enums), 45
- patch() (sevenbridges.http.client.HttpClient method), 32
- path (sevenbridges.transfer.download.Download attribute), 54
- pause() (sevenbridges.transfer.download.Download method), 55
- pause() (sevenbridges.transfer.upload.Upload method), 55
- PAUSED (sevenbridges.models.enums.TransferState attribute), 46
- PAUSED (sevenbridges.TransferState attribute), 30
- pending (sevenbridges.BillingGroup attribute), 22
- PENDING (sevenbridges.ImportExportState attribute), 46
- pending (sevenbridges.Invoice attribute), 22
- pending (sevenbridges.models.billing_group.BillingGroup attribute), 44

- PENDING (sevenbridges.models.enums.ImportExportState attribute), 45
- pending (sevenbridges.models.invoice.Invoice attribute), 48
- Permissions (class in sevenbridges), 27
- Permissions (class in sevenbridges.models.compound.projects.permissions), 39
- permissions (sevenbridges.Member attribute), 27
- permissions (sevenbridges.models.member.Member attribute), 48
- phone (sevenbridges.models.user.User attribute), 53
- phone (sevenbridges.User attribute), 23
- PLATFORM (sevenbridges.FileStorageType attribute), 31
- PLATFORM (sevenbridges.models.enums.FileStorageType attribute), 45
- post() (sevenbridges.http.client.HttpClient method), 32
- PREPARING (sevenbridges.models.enums.TransferState attribute), 46
- PREPARING (sevenbridges.TransferState attribute), 30
- previous_page() (sevenbridges.meta.collection.Collection method), 33
- Price (class in sevenbridges.models.compound.price), 42
- price (sevenbridges.models.task.Task attribute), 52
- price (sevenbridges.Task attribute), 25
- PROBLEM (sevenbridges.models.enums.FeedbackType attribute), 45
- Profile (class in sevenbridges.config), 57
- Progress (class in sevenbridges.transfer.utils), 56
- progress (sevenbridges.transfer.download.Download attribute), 55
- progress (sevenbridges.transfer.upload.Upload attribute), 55
- progress (sevenbridges.transfer.utils.Progress attribute), 56
- Project (class in sevenbridges), 23
- Project (class in sevenbridges.models.project), 49
- project (sevenbridges.App attribute), 26
- project (sevenbridges.File attribute), 27
- project (sevenbridges.models.app.App attribute), 43
- project (sevenbridges.models.compound.volumes.import_destination.ImportDestination attribute), 41
- project (sevenbridges.models.file.File attribute), 47
- project (sevenbridges.models.task.Task attribute), 52
- project (sevenbridges.Task attribute), 25
- project_breakdown (sevenbridges.BillingGroupBreakdown attribute), 22
- project_breakdown (sevenbridges.models.billing_breakdown.BillingGroupBreakdown attribute), 43
- ProjectBreakdown (class in sevenbridges.models.compound.billing.project_breakdown), 22
- projects (sevenbridges.Api attribute), 21
- projects (sevenbridges.api.Api attribute), 57
- projects_url (sevenbridges.Endpoints attribute), 23
- projects_url (sevenbridges.models.endpoints.Endpoints attribute), 44
- properties (sevenbridges.Export attribute), 28
- properties (sevenbridges.models.storage_export.Export attribute), 50
- provider (sevenbridges.models.compound.jobs.job_instance.Instance attribute), 38
- PROXIES (sevenbridges.config.Profile attribute), 57
- proxies (sevenbridges.config.Profile attribute), 57
- put() (sevenbridges.http.client.HttpClient method), 32
- ## Q
- query() (sevenbridges.App class method), 26
- query() (sevenbridges.BillingGroup class method), 22
- query() (sevenbridges.Export class method), 28
- query() (sevenbridges.File class method), 27
- query() (sevenbridges.Import class method), 29
- query() (sevenbridges.Invoice class method), 22
- query() (sevenbridges.models.app.App class method), 43
- query() (sevenbridges.models.billing_group.BillingGroup class method), 44
- query() (sevenbridges.models.file.File class method), 47
- query() (sevenbridges.models.invoice.Invoice class method), 48
- query() (sevenbridges.models.project.Project class method), 50
- query() (sevenbridges.models.storage_export.Export class method), 50
- query() (sevenbridges.models.storage_import.Import class method), 51
- query() (sevenbridges.models.task.Task class method), 52
- query() (sevenbridges.models.volume.Volume class method), 54
- query() (sevenbridges.Project class method), 24
- query() (sevenbridges.Task class method), 25
- query() (sevenbridges.Volume class method), 30
- queued (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
- QUEUED (sevenbridges.models.enums.TaskStatus attribute), 45
- QUEUED (sevenbridges.TaskStatus attribute), 31
- queued_duration (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
- ## R
- rate_limit (sevenbridges.Api attribute), 21
- rate_limit (sevenbridges.api.Api attribute), 57
- rate_limit_sleeper() (in module sevenbridges.http.error_handlers), 33

- rate_limit_url (sevenbridges.Endpoints attribute), 23
 - rate_limit_url (sevenbridges.models.endpoints.Endpoints attribute), 44
 - raw (sevenbridges.App attribute), 26
 - raw (sevenbridges.models.app.App attribute), 43
 - READ_ONLY (sevenbridges.models.enums.VolumeAccessMode attribute), 46
 - READ_ONLY (sevenbridges.VolumeAccessMode attribute), 30
 - READ_WRITE (sevenbridges.models.enums.VolumeAccessMode attribute), 46
 - READ_WRITE (sevenbridges.VolumeAccessMode attribute), 30
 - ReadOnlyPropertyError, 31, 58
 - rel (sevenbridges.models.link.Link attribute), 48
 - reload() (sevenbridges.File method), 27
 - reload() (sevenbridges.meta.resource.Resource method), 35
 - reload() (sevenbridges.models.file.File method), 47
 - remaining (sevenbridges.http.client.HttpClient attribute), 32
 - remove_error_handler() (sevenbridges.http.client.HttpClient method), 32
 - remove_member() (sevenbridges.models.project.Project method), 50
 - remove_member() (sevenbridges.Project method), 24
 - request_id (sevenbridges.http.client.HttpClient attribute), 32
 - RequestTimeout, 32, 58
 - reset_time (sevenbridges.http.client.HttpClient attribute), 33
 - Resource (class in sevenbridges.meta.resource), 35
 - resource (sevenbridges.meta.collection.Collection attribute), 33
 - ResourceMeta (class in sevenbridges.meta.resource), 35
 - ResourceNotModified, 31, 59
 - result (sevenbridges.Export attribute), 28
 - result (sevenbridges.Import attribute), 29
 - result (sevenbridges.models.storage_export.Export attribute), 50
 - result (sevenbridges.models.storage_import.Import attribute), 51
 - result() (sevenbridges.transfer.upload.Upload method), 55
 - resume() (sevenbridges.transfer.download.Download method), 55
 - resume() (sevenbridges.transfer.upload.Upload method), 56
 - retried (sevenbridges.models.compound.jobs.job.Job attribute), 38
 - retry() (in module sevenbridges.decorators), 58
 - retry_on_exc() (in module sevenbridges.decorators), 58
 - revision (sevenbridges.App attribute), 26
 - revision (sevenbridges.models.app.App attribute), 43
 - run() (sevenbridges.models.task.Task method), 52
 - run() (sevenbridges.Task method), 26
 - run() (sevenbridges.transfer.download.Download method), 55
 - run() (sevenbridges.transfer.upload.Upload method), 56
 - Member_username (sevenbridges.models.compound.billing.task_breakdown.TaskBreakdown attribute), 36
 - RUNNING (sevenbridges.ImportExportState attribute), 31
 - running (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
 - RUNNING (sevenbridges.models.enums.ImportExportState attribute), 45
 - RUNNING (sevenbridges.models.enums.TaskStatus attribute), 45
 - RUNNING (sevenbridges.models.enums.TransferState attribute), 46
 - RUNNING (sevenbridges.TaskStatus attribute), 31
 - RUNNING (sevenbridges.TransferState attribute), 30
 - running_duration (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
- ## S
- S3 (sevenbridges.models.enums.VolumeType attribute), 46
 - S3 (sevenbridges.VolumeType attribute), 30
 - save() (sevenbridges.File method), 28
 - save() (sevenbridges.Member method), 27
 - save() (sevenbridges.models.file.File method), 47
 - save() (sevenbridges.models.member.Member method), 48
 - save() (sevenbridges.models.project.Project method), 50
 - save() (sevenbridges.models.task.Task method), 52
 - save() (sevenbridges.models.volume.Volume method), 54
 - save() (sevenbridges.Project method), 24
 - save() (sevenbridges.Task method), 26
 - save() (sevenbridges.Volume method), 30
 - SbgError, 31, 59
 - ServerError, 32, 59
 - service (sevenbridges.models.volume.Volume attribute), 54
 - service (sevenbridges.Volume attribute), 30
 - ServiceUnavailable, 32, 59
 - session (sevenbridges.http.client.HttpClient attribute), 33
 - Settings (class in sevenbridges.models.compound.projects.settings), 39
 - settings (sevenbridges.models.project.Project attribute), 50
 - settings (sevenbridges.Project attribute), 24
 - sevenbridges (module), 21
 - sevenbridges.api (module), 56

- sevenbridges.config (module), 57
- sevenbridges.decorators (module), 58
- sevenbridges.errors (module), 58
- sevenbridges.http (module), 32
- sevenbridges.http.client (module), 32
- sevenbridges.http.error_handlers (module), 33
- sevenbridges.meta (module), 33
- sevenbridges.meta.collection (module), 33
- sevenbridges.meta.comp_mutable_dict (module), 33
- sevenbridges.meta.data (module), 34
- sevenbridges.meta.fields (module), 34
- sevenbridges.meta.resource (module), 35
- sevenbridges.meta.transformer (module), 35
- sevenbridges.models (module), 36
- sevenbridges.models.app (module), 43
- sevenbridges.models.billing_breakdown (module), 43
- sevenbridges.models.billing_group (module), 44
- sevenbridges.models.compound (module), 36
- sevenbridges.models.compound.billing (module), 36
- sevenbridges.models.compound.billing.invoice_period (module), 36
- sevenbridges.models.compound.billing.project_breakdown (module), 36
- sevenbridges.models.compound.billing.task_breakdown (module), 36
- sevenbridges.models.compound.error (module), 42
- sevenbridges.models.compound.files (module), 37
- sevenbridges.models.compound.files.download_info (module), 37
- sevenbridges.models.compound.files.file_origin (module), 37
- sevenbridges.models.compound.files.file_storage (module), 37
- sevenbridges.models.compound.files.metadata (module), 37
- sevenbridges.models.compound.jobs (module), 38
- sevenbridges.models.compound.jobs.job (module), 38
- sevenbridges.models.compound.jobs.job_docker (module), 38
- sevenbridges.models.compound.jobs.job_instance (module), 38
- sevenbridges.models.compound.jobs.job_log (module), 39
- sevenbridges.models.compound.price (module), 42
- sevenbridges.models.compound.projects (module), 39
- sevenbridges.models.compound.projects.permissions (module), 39
- sevenbridges.models.compound.projects.settings (module), 39
- sevenbridges.models.compound.tasks (module), 39
- sevenbridges.models.compound.tasks.batch_by (module), 39
- sevenbridges.models.compound.tasks.batch_group (module), 40
- sevenbridges.models.compound.tasks.execution_status (module), 40
- sevenbridges.models.compound.tasks.input (module), 41
- sevenbridges.models.compound.tasks.output (module), 41
- sevenbridges.models.compound.volumes (module), 41
- sevenbridges.models.compound.volumes.credentials (module), 41
- sevenbridges.models.compound.volumes.import_destination (module), 41
- sevenbridges.models.compound.volumes.properties (module), 41
- sevenbridges.models.compound.volumes.service (module), 42
- sevenbridges.models.compound.volumes.volume_file (module), 42
- sevenbridges.models.endpoints (module), 44
- sevenbridges.models.enums (module), 45
- sevenbridges.models.execution_details (module), 46
- sevenbridges.models.file (module), 46
- sevenbridges.models.invoice (module), 48
- sevenbridges.models.link (module), 48
- sevenbridges.models.member (module), 48
- sevenbridges.models.project (module), 49
- sevenbridges.models.storage_export (module), 50
- sevenbridges.models.storage_import (module), 51
- sevenbridges.models.task (module), 51
- sevenbridges.models.user (module), 53
- sevenbridges.models.volume (module), 53
- sevenbridges.transfer (module), 54
- sevenbridges.transfer.download (module), 54
- sevenbridges.transfer.upload (module), 55
- sevenbridges.transfer.utils (module), 56
- simple_progress_bar() (in module sevenbridges.transfer.utils), 56
- size (sevenbridges.File attribute), 28
- size (sevenbridges.models.file.File attribute), 47
- size (sevenbridges.transfer.utils.Part attribute), 56
- source (sevenbridges.Export attribute), 28
- source (sevenbridges.Import attribute), 29
- source (sevenbridges.models.storage_export.Export attribute), 50
- source (sevenbridges.models.storage_import.Import attribute), 51
- start (sevenbridges.transfer.utils.Part attribute), 56
- start() (sevenbridges.transfer.download.Download method), 55
- start() (sevenbridges.transfer.upload.Upload method), 56
- start_time (sevenbridges.models.compound.jobs.job.Job attribute), 38
- start_time (sevenbridges.models.execution_details.ExecutionDetails attribute), 46
- start_time (sevenbridges.models.task.Task attribute), 52
- start_time (sevenbridges.Task attribute), 26

- start_time (sevenbridges.transfer.download.Download attribute), 55
 - start_time (sevenbridges.transfer.upload.Upload attribute), 56
 - started_on (sevenbridges.Export attribute), 28
 - started_on (sevenbridges.Import attribute), 29
 - started_on (sevenbridges.models.storage_export.Export attribute), 50
 - started_on (sevenbridges.models.storage_import.Import attribute), 51
 - state (sevenbridges.Export attribute), 29
 - state (sevenbridges.Import attribute), 29
 - state (sevenbridges.models.storage_export.Export attribute), 50
 - state (sevenbridges.models.storage_import.Import attribute), 51
 - state (sevenbridges.models.user.User attribute), 53
 - state (sevenbridges.User attribute), 23
 - status (sevenbridges.models.compound.error.Error attribute), 42
 - status (sevenbridges.models.compound.jobs.job.Job attribute), 38
 - status (sevenbridges.models.execution_details.ExecutionDetails attribute), 46
 - status (sevenbridges.models.task.Task attribute), 52
 - status (sevenbridges.Task attribute), 26
 - status (sevenbridges.transfer.download.Download attribute), 55
 - status (sevenbridges.transfer.upload.Upload attribute), 56
 - steps_completed (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
 - steps_total (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
 - stop() (sevenbridges.transfer.download.Download method), 55
 - stop() (sevenbridges.transfer.upload.Upload method), 56
 - STOPPED (sevenbridges.models.enums.TransferState attribute), 46
 - STOPPED (sevenbridges.TransferState attribute), 30
 - storage (sevenbridges.File attribute), 28
 - storage (sevenbridges.models.file.File attribute), 47
 - storage_costs (sevenbridges.Invoice attribute), 22
 - storage_costs (sevenbridges.models.invoice.Invoice attribute), 48
 - stream() (sevenbridges.File method), 28
 - stream() (sevenbridges.models.file.File method), 47
 - StringField (class in sevenbridges.meta.fields), 34
 - submit() (sevenbridges.transfer.download.DPartedFile method), 54
 - submit() (sevenbridges.transfer.upload.UPartedFile method), 55
 - submit_export() (sevenbridges.Export class method), 29
 - submit_export() (sevenbridges.models.storage_export.Export class method), 50
 - submit_import() (sevenbridges.Import class method), 29
 - submit_import() (sevenbridges.models.storage_import.Import class method), 51
 - system_limit (sevenbridges.models.compound.tasks.execution_status.ExecutionStatus attribute), 40
- ## T
- tags (sevenbridges.File attribute), 28
 - tags (sevenbridges.models.file.File attribute), 47
 - tags (sevenbridges.models.project.Project attribute), 50
 - tags (sevenbridges.Project attribute), 24
 - Task (class in sevenbridges), 25
 - Task (class in sevenbridges.models.task), 51
 - task (sevenbridges.models.compound.files.file_origin.FileOrigin attribute), 37
 - task_breakdown (sevenbridges.models.compound.billing.project_breakdown.ProjectBreakdown attribute), 36
 - task_cost (sevenbridges.models.compound.billing.task_breakdown.TaskBreakdown attribute), 36
 - TaskBreakdown (class in sevenbridges.models.compound.billing.task_breakdown), 36
 - tasks (sevenbridges.Api attribute), 21
 - tasks (sevenbridges.api.Api attribute), 57
 - tasks_url (sevenbridges.Endpoints attribute), 23
 - tasks_url (sevenbridges.models.endpoints.Endpoints attribute), 44
 - TaskStatus (class in sevenbridges), 31
 - TaskStatus (class in sevenbridges.models.enums), 45
 - TaskValidationError, 31, 59
 - TB (sevenbridges.models.enums.PartSize attribute), 45
 - THOUGHT (sevenbridges.models.enums.FeedbackType attribute), 45
 - time_finished (sevenbridges.models.compound.billing.task_breakdown.TaskBreakdown attribute), 36
 - time_started (sevenbridges.models.compound.billing.task_breakdown.TaskBreakdown attribute), 36
 - to (sevenbridges.models.compound.billing.invoice_period.InvoicePeriod attribute), 36
 - to_app() (sevenbridges.meta.transformer.Transform static method), 35
 - to_billing_group() (sevenbridges.meta.transformer.Transform static method), 35
 - to_datestring() (sevenbridges.meta.transformer.Transform static method), 35
 - to_file() (sevenbridges.meta.transformer.Transform static method), 35
 - to_project() (sevenbridges.meta.transformer.Transform static method), 35

to_task() (sevenbridges.meta.transformer.Transform static method), 35

to_user() (sevenbridges.meta.transformer.Transform static method), 35

to_volume() (sevenbridges.meta.transformer.Transform static method), 35

TooManyRequests, 32, 59

total (sevenbridges.Invoice attribute), 22

total (sevenbridges.meta.collection.Collection attribute), 33

total (sevenbridges.models.invoice.Invoice attribute), 48

total_parts() (in module sevenbridges.transfer.utils), 56

total_spending (sevenbridges.BillingGroupBreakdown attribute), 22

total_spending (sevenbridges.models.billing_breakdown.BillingGroupBreakdown attribute), 43

TransferState (class in sevenbridges), 30

TransferState (class in sevenbridges.models.enums), 45

Transform (class in sevenbridges.meta.transformer), 35

type (sevenbridges.BillingGroup attribute), 22

type (sevenbridges.models.billing_group.BillingGroup attribute), 44

type (sevenbridges.models.compound.files.file_storage.FileStorage attribute), 37

type (sevenbridges.models.compound.jobs.job_instance.Instance attribute), 38

type (sevenbridges.models.project.Project attribute), 50

type (sevenbridges.models.task.Task attribute), 52

type (sevenbridges.Project attribute), 25

type (sevenbridges.Task attribute), 26

U

Unauthorized, 31, 59

UPartedFile (class in sevenbridges.transfer.upload), 55

update() (sevenbridges.meta.comp_mutable_dict.CompoundMutableDict method), 34

update() (sevenbridges.models.compound.tasks.batch_by.BatchBy method), 39

Upload (class in sevenbridges.transfer.upload), 55

upload() (sevenbridges.File class method), 28

upload() (sevenbridges.models.file.File class method), 47

UPLOAD_MINIMUM_PART_SIZE (sevenbridges.models.enums.PartSize attribute), 45

upload_url (sevenbridges.Endpoints attribute), 23

upload_url (sevenbridges.models.endpoints.Endpoints attribute), 44

url (sevenbridges.models.compound.files.download_info.DownloadInfo attribute), 37

User (class in sevenbridges), 22

User (class in sevenbridges.models.user), 53

user_url (sevenbridges.Endpoints attribute), 23

user_url (sevenbridges.models.endpoints.Endpoints attribute), 44

username (sevenbridges.Member attribute), 27

username (sevenbridges.models.member.Member attribute), 49

username (sevenbridges.models.user.User attribute), 53

username (sevenbridges.User attribute), 23

users (sevenbridges.Api attribute), 21

users (sevenbridges.api.Api attribute), 57

users_url (sevenbridges.Endpoints attribute), 23

users_url (sevenbridges.models.endpoints.Endpoints attribute), 44

UuidField (class in sevenbridges.meta.fields), 35

V

validate() (sevenbridges.meta.fields.BasicListField method), 34

validate() (sevenbridges.meta.fields.BooleanField method), 34

validate() (sevenbridges.meta.fields.Field method), 34

validate() (sevenbridges.meta.fields.FloatField method), 34

validate() (sevenbridges.meta.fields.IntegerField method), 34

validate() (sevenbridges.meta.fields.StringField method), 35

validate() (sevenbridges.meta.fields.UuidField method), 35

ValidationError, 31, 59

value (sevenbridges.models.compound.tasks.batch_group.BatchGroup attribute), 40

Volume (class in sevenbridges), 29

Volume (class in sevenbridges.models.volume), 53

VOLUME (sevenbridges.FileStorageType attribute), 31

volume (sevenbridges.models.compound.files.file_storage.FileStorage attribute), 37

volume (sevenbridges.models.compound.volumes.volume_file.VolumeFile attribute), 42

VOLUME (sevenbridges.models.enums.FileStorageType attribute), 45

VolumeAccessMode (class in sevenbridges), 30

VolumeAccessMode (class in sevenbridges.models.enums), 46

VolumeCredentials (class in sevenbridges.models.compound.volumes.credentials), 41

VolumeFile (class in sevenbridges.models.compound.volumes.volume_file), 42

VolumeProperties (class in sevenbridges.models.compound.volumes.properties), 41

volumes (sevenbridges.Api attribute), 21

volumes (sevenbridges.api.Api attribute), 57

VolumeService (class in sevenbridges.models.compound.volumes.service),

42

VolumeType (class in sevenbridges), 30

VolumeType (class in sevenbridges.models.enums), 46

W

wait() (sevenbridges.transfer.download.Download method), 55

wait() (sevenbridges.transfer.upload.Upload method), 56

warnings (sevenbridges.models.task.Task attribute), 52

warnings (sevenbridges.Task attribute), 26

Z

zip_code (sevenbridges.models.user.User attribute), 53

zip_code (sevenbridges.User attribute), 23