

# Package ‘sbgr’

April 23, 2016

**Type** Package

**Title** R Client for Seven Bridges Genomics API

**Version** 1.0.0

**Date** 2015-05-01

**Author** Nan Xiao <road2stat@gmail.com>, Tengfei Yin <tengfei.yin@sbgenomics.com>

**Maintainer** Nan Xiao <road2stat@gmail.com>

**Description** R client for Seven Bridges Genomics API.

**License** MIT + file LICENSE

**VignetteBuilder** knitr

**URL** <https://www.sbgenomics.com>

**BugReports** <https://github.com/road2stat/sbgr/issues>

**biocViews** Software, DataImport, ThirdPartyClient

**Depends** methods, utils, stats

**Imports** htr, jsonlite, objectProperties,

**Suggests** BiocStyle, knitr, rmarkdown, testthat

**NeedsCompilation** no

## R topics documented:

sbgr-package . . . . .	2
Auth-class . . . . .	3
billing . . . . .	4
FileTypeSingleEnum-class . . . . .	5
file_copy . . . . .	6
file_delete . . . . .	7
file_details . . . . .	7
file_download_url . . . . .	8
file_list . . . . .	9
file_meta_update . . . . .	9
Item-class . . . . .	11

misc_get_auth_token . . . . .	11
misc_get_uploader . . . . .	12
misc_make_metadata . . . . .	12
misc_upload_cli . . . . .	14
pipeline_add . . . . .	15
pipeline_details . . . . .	16
pipeline_list_my . . . . .	17
pipeline_list_project . . . . .	17
pipeline_list_pub . . . . .	18
project_delete . . . . .	18
project_details . . . . .	19
project_list . . . . .	20
project_members . . . . .	20
project_member_add . . . . .	21
project_member_delete . . . . .	22
project_member_update . . . . .	22
project_new . . . . .	23
sbgapi . . . . .	24
Task-class . . . . .	25
task_action . . . . .	27
task_details . . . . .	28
task_list . . . . .	29
task_run . . . . .	29
upload_complete_all . . . . .	31
upload_complete_part . . . . .	31
upload_delete . . . . .	32
upload_info . . . . .	33
upload_info_part . . . . .	33
upload_init . . . . .	34
<b>Index</b>	<b>36</b>

---

 sbgr-package

*R Client for Seven Bridges Genomics API*


---

## Description

R Client for Seven Bridges Genomics API

## Details

The vignette can be opened with `vignette('sbgr')`.

Package: sbgr  
 Type: Package  
 License: MIT

**Author(s)**

Nan Xiao <<nan.xiao@sbgenomics.com>> Teng-Fei Yin <<tengfei.yin@sbgenomics.com>>

---

 Auth-class

*Class Auth*


---

**Description**

Auth token object

**Arguments**

auth_token	[character] your auth token.
api	[character using, by default it is sbg us platform.
url	[chracter] a URL for the API, default is NULL, will use api parameter to switch to the right one.
version	[character] default: 1.1 version used for api.

**Details**

Every object could be requested from this Auth object and any action could start from this object using cascading style. Please check vignette 'easy-api' for more information.

**Fields**

auth\_token [character] your auth token.  
 url [character] basic url used for API, by default it's <https://api.sbgenomics.com/1.1/>

**Methods**

```
project(name = NULL, id = NULL, index = NULL, ignore.case = TRUE, exact = TRUE)
  find project
```

**Examples**

```
## replace it with real token
token <- "aef7e9e3f6c54fb1b338ac4ecddf1a56"
a <- Auth(token)

## get billing info
b <- a$billing()
## create project
a$project_new(name = "API", description = "API tutorial",
              billing_group_id = b[[1]]$id)
p <- a$project("API")
## get data
fl <- system.file("extdata", "sample1.fastq", package = "sbgr")
```

```

## create meta data
fl.meta <- list(file_type = "fastq",
               seq_tech = "Illumina",
               sample = "sample1",
               author = "tengfei")

## upload data with metadata
p$upload(fl, metadata = fl.meta)
## check uploading success
f.file <- p$file(basename(fl))
## get the pipeline from public repos
f.pipe <- a$pipeline(pipeline_name = "FastQC")
## copy the pipeline to your project
p$pipeline_add(pipeline_name = f.pipe$name)
## get the pipeline from your project not public one
f.pipe <- p$pipeline(name = "FastQC")
## check the inputs needed for running tasks
f.pipe$details()
## Ready to run a task? go
f.task <- p$task_run(name = "my task",
                    description = "A text description",
                    pipeline_id = f.pipe$id,
                    inputs = list(
                        "177252" = list(f.file$id)
                    ))

f.task$run()
## or you can just run with Task constructor
f.task <- Task(auth = Auth(token),
              name = "my task",
              description = "A text description",
              pipeline_id = f.pipe$id,
              project_id = p$id,
              inputs = list(
                  "177252" = list(f.file$id)
              ))
## Monitor you task
f.task$monitor(30)

## download a task output files
f.task <- p$task("my task")
f.task$download("~/Desktop/")

## Abort the task
f.task$abort()

```

**Description**

Returns the list of all billing groups you have access to. This is an utility method used only create projects with appropriate billing group. Full access to billing data is not available via the SBG public API yet.

**Usage**

```
billing(auth_token = NULL, ...)
```

**Arguments**

auth_token	auth token
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'  
req = billing(token)
```

---

FileTypeSingleEnum-class

*Metadata class*

---

**Description**

Metadata class

**Details**

This function will help you create a Metadata object, what it does it to accept a named list or just pass meta key-value pairs as argument one by one. Then it first matches SBG's build-in meta field which will be shown in the graphic interface on the platform, then save extra meta information in extra field, but not visible on the platform yet, you can view it via the API.

There are four pre-defined fields with pre-defined levels, they are file\_type, qual\_scale, seq\_tech, and paired\_end, those are also constructor names to construct a single Enum object, it's different from characters, it has validation against levels, to check their levels, you can simply create a empty Metadata object and access the field levels. Please see examples.

**References**

<https://docs.sbggenomics.com/display/sbg/Metadata>

**Examples**

```
m <- Metadata()
## to check levels
m$file_type
levels(m$file_type)
## to replace a Enum class need to use constructor
m$file_type <- file_type("text")
```

---

file\_copy

*Copy specified file(s) to the specified project*


---

**Description**

Copy specified file(s) to the specified project

**Usage**

```
file_copy(auth_token = NULL, project_id = NULL, file_id = NULL, ...)
```

**Arguments**

auth_token	auth token
project_id	ID of a project you want to copy files to.
file_id	Character vector. IDs of the files you wish to copy to the project.
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'
req = file_copy(token,
  project_id = '7f7a72d0-da77-4f51-9416-99f14f7316ab',
  file_id = c('5506a44ae4b04a4ab3ae7250',
    '5506a44ae4b04a4ab3ae7254',
    '5506a44ae4b04a4ab3ae7252'))
```

---

file_delete	<i>Removes a file from a project</i>
-------------	--------------------------------------

---

**Description**

Removes a file from a project

**Usage**

```
file_delete(auth_token = NULL, project_id = NULL, file_id = NULL, ...)
```

**Arguments**

auth_token	auth token
project_id	ID of a project you want to access.
file_id	ID of a file you want to delete.
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'  
req = file_delete(token,  
                  project_id = '1c1d06d2-5862-48f6-b595-e0099b20937e',  
                  file_id = '530854e2e4b036506b803c7e')
```

---

file_details	<i>Returns detailed information about a project's files</i>
--------------	---

---

**Description**

Returns detailed information about a project's files

**Usage**

```
file_details(auth_token = NULL, project_id = NULL, file_id = NULL, ...)
```

**Arguments**

auth_token	auth token
project_id	ID of a project you want to access.
file_id	ID of a file you want to access.
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'  
req = file_details(token,  
    project_id = '1c1d06d2-5862-48f6-b595-e0099b20937e',  
    file_id = '530854e2e4b036506b803c7e')
```

---

file_download_url	<i>Returns a direct download URL for a project's file</i>
-------------------	---

---

**Description**

Returns a direct download URL for a project's file.

**Usage**

```
file_download_url(auth_token = NULL, project_id = NULL, file_id = NULL,  
    ...)
```

**Arguments**

auth_token	auth token
project_id	ID of a project you want to access.
file_id	ID of a file you want to access.
...	parameters passed to sbgapi function

**Details**

You can use any HTTP client, or library to access or download the content once you get the URL.

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'  
req = file_download_url(token,  
    project_id = '1c1d06d2-5862-48f6-b595-e0099b20937e',  
    file_id = '530854e2e4b036506b803c7e')
```



---

file_list	<i>Returns the list of all project files for a project</i>
-----------	--

---

### Description

Returns the list of all project files for a project. If user specifies string "public" as project\_id, this will return a list of public files.

### Usage

```
file_list(auth_token = NULL, project_id = NULL, ...)
```

### Arguments

auth_token	auth token
project_id	ID of a project you want to access. Note that specifying "public" you can list public files.
...	parameters passed to sbgapi function

### Value

parsed list of the returned json

### Examples

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'
req = file_list(token,
                project_id = '1c1d06d2-5862-48f6-b595-e0099b20937e')
```

---

file_meta_update	<i>Update project's file metadata</i>
------------------	---------------------------------------

---

### Description

This function updates project's file metadata. You can also use this call to change filenames if you supply the name argument.

### Usage

```
file_meta_update(auth_token = NULL, project_id = NULL, file_id = NULL,
                 name = NULL, file_type = c("text", "binary", "fasta", "csfasta", "fastq",
                 "qual", "xsq", "sff", "bam", "bam_index", "illumina_export", "vcf", "sam",
                 "bed", "archive", "juncs", "gtf", "gff", "enlis_genome"),
                 qual_scale = c("sanger", "illumina13", "illumina15", "illumina18",
                 "solexa"), seq_tech = c("454", "Helicos", "Illumina", "Solid",
                 "IonTorrent"), sample = NULL, library = NULL, platform_unit = NULL,
                 paired_end = NULL, ...)
```

**Arguments**

auth_token	auth token
project_id	ID of a project you want to access.
file_id	ID of a file you want to access.
name	File name.
file_type	File type. This metadata parameter is mandatory for each file.
qual_scale	Quality scale encoding. For FASTQ files, you must either specify the quality score encoding sch which contains the FASTQ quality scale detector wrapper. In that case, you can specify the quality score encoding scheme by setting qual_scale inside the pipeline. For BAM files, this value should always be 'sanger'.
seq_tech	Sequencing technology. The seq_tech parameter allows you to specify the sequencing technology used. This metadata parameter is only required by some the tools and pipelines; however, it is strongly recommended that you set it whenever possible, unless you are certain that your pipeline will work without it.
sample	Sample ID. You can use the sample parameter to specify the sample identifier. The value supplied in this field will be written to the read group tag (@RG:SM) in SAM/BAM files generated from reads with the specified Sample ID. AddOrReplaceReadGroups will use this parameter as the value for the read group tag in a SAM/BAM file.
library	Library. You can set the library for the read using the library parameter. The value supplied in this field will be written to the read group tag (@RG:LB) in SAM/BAM files generated from reads with the specified Library ID. AddOrReplaceReadGroups will use this parameter as the value for the read group tag in a SAM/BAM file.
platform_unit	Platform unit. You can set the platform unit (e.g. lane for Illumina, or slide for SOLiD) using the platform_unit parameter. The value supplied in this field will be written to the read group tag (@RG:PU) in SAM/BAM files generated from the reads with the specified Platform Unit. AddOrReplaceReadGroups will use this parameter as the value for the read group tag of a SAM/BAM file.
paired_end	Paired end. With paired-end reads, this parameter indicates if the read file is left end (1) or right end (2). For SOLiD CSFASTA files, paired end files 1 and 2 correspond to R3 and F3 files, respectively.
...	parameters passed to sbgapi function

**Details**

For more information about file metadata, please check the File Metadata Documentation: <https://developer.sbgenomics.com/platform/metadata>.

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'
req = file_meta_update(token,
  project_id = '1c1d06d2-5862-48f6-b595-e0099b20937e',
  file_id = '530854e2e4b036506b803c7e',
  name = 'c.elegans_chr2_test.fastq',
  file_type = 'fastq', qual_scale = 'illumina13',
  seq_tech = 'Illumina')
```

---

Item-class

*Class Item*


---

**Description**

Class Item

**Details**

To describe a set of objects, Project, Task, Pipeline, File etc.

**Fields**

response save the raw response from a request.

auth\_token propagate the auth\_token from parent.

---

misc\_get\_auth\_token

*Opens browser to copy the auth token*


---

**Description**

Click 'Generate Token' button, copy and paste the generated token string to the R console. The function will return the token string.

**Usage**

```
misc_get_auth_token()
```

**Value**

auth token

**Examples**

```
# Paste the auth token into R
# console then press enter:
token = NULL
token = misc_get_auth_token()
```

---

misc_get_uploader	<i>Download SBG uploader and extract to a specified directory</i>
-------------------	---

---

**Description**

Download SBG uploader and extract to a specified directory.

**Usage**

```
misc_get_uploader(destdir = NULL)
```

**Arguments**

destdir	The directory to extract SBG uploader to. If not present, it will be created automatically.
---------	---

**Value**

∅L if the SBG CLI uploader is successfully downloaded and unarchived.

**Examples**

```
dir = '~/sbg-uploader/'
misc_get_uploader(dir)
```

---

misc_make_metadata	<i>Specify the parameters of the file metadata and return a list, JSON string, or write to a file</i>
--------------------	---

---

**Description**

Specify the parameters of the file metadata and return a list, JSON string, or write to a file.

**Usage**

```
misc_make_metadata(output = c("list", "json", "metafile"), destfile = NULL,
  name = NULL, file_type = c("text", "binary", "fasta", "csfasta", "fastq",
  "qual", "xsq", "sff", "bam", "bam_index", "illumina_export", "vcf", "sam",
  "bed", "archive", "juncs", "gtf", "gff", "enlis_genome"),
  qual_scale = c("sanger", "illumina13", "illumina15", "illumina18",
  "solexa"), seq_tech = c("454", "Helicos", "Illumina", "Solid",
  "IonTorrent"), sample = NULL, library = NULL, platform_unit = NULL,
  paired_end = NULL)
```

## Arguments

output	Output format, could be 'list', 'json', or 'metafile'.
destfile	Filename to write to. Must be specified when output = 'metafile'.
name	File name.
file_type	File type. This metadata parameter is mandatory for each file.
qual_scale	Quality scale encoding. For FASTQ files, you must either specify the quality score encoding sch which contains the FASTQ quality scale detector wrapper. In that case, you can specify the quality score encoding scheme by setting qual_scale inside the pipeline. For BAM files, this value should always be 'sanger'.
seq_tech	Sequencing technology. The seq_tech parameter allows you to specify the sequencing technology used. This metadata parameter is only required by some the tools and pipelines; however, it is strongly recommended that you set it whenever possible, unless you are certain that your pipeline will work without it.
sample	Sample ID. You can use the sample parameter to specify the sample identifier. The value supplied in this field will be written to the read group tag (@RG:SM) in SAM/BAM files generated from reads with the specified Sample ID. AddOrReplaceReadGroups will use this parameter as the value for the read group tag in a SAM/BAM file.
library	Library. You can set the library for the read using the library parameter. The value supplied in this field will be written to the read group tag (@RG:LB) in SAM/BAM files generated from reads with the specified Library ID. AddOrReplaceReadGroups will use this parameter as the value for the read group tag in a SAM/BAM file.
platform_unit	Platform unit. You can set the platform unit (e.g. lane for Illumina, or slide for SOLiD) using the platform_unit parameter. The value supplied in this field will be written to the read group tag (@RG:PU) in SAM/BAM files generated from the reads with the specified Platform Unit. AddOrReplaceReadGroups will use this parameter as the value for the read group tag of a SAM/BAM file.
paired_end	Paired end. With paired-end reads, this parameter indicates if the read file is left end (1) or right end (2). For SOLiD CSFASTA files, paired end files 1 and 2 correspond to R3 and F3 files, respectively.

## Details

For more information about file metadata, please check the File Metadata Documentation: <https://developer.sbgenomics.com/platform/metadata>.

## Value

list, JSON string, or a file.

## References

<https://developer.sbgenomics.com/platform/metadata>

**Examples**

```

destfile = '~/c.elegans_chr2_test.fastq.meta'
misc_make_metadata(output = 'metafile',
                  destfile = destfile,
                  name = 'c.elegans_chr2_test.fastq',
                  file_type = 'fastq', qual_scale = 'illumina13',
                  seq_tech = 'Illumina')

```

---

misc_upload_cli	<i>Upload files using SBG uploader</i>
-----------------	--

---

**Description**

Upload files using SBG uploader.

**Usage**

```

misc_upload_cli(auth_token = NULL, uploader = NULL, file = NULL,
               project_id = NULL, proxy = NULL)

```

**Arguments**

auth_token	auth token
uploader	The directory where the SBG uploader is located (the directory that contains the bin/ directory).
file	The location of the file to upload.
project_id	The project ID to upload the files to. If you do not supply this, then the uploader will place the incoming files in your "My Files" section.
proxy	Allows you to specify a proxy server through which the uploader should connect. About the details the proxy parameter format, see <a href="https://developer.sbgenomics.com/tools/uploader/documentation">https://developer.sbgenomics.com/tools/uploader/documentation</a> .

**Value**

The uploaded file's ID number.

**References**

<https://developer.sbgenomics.com/tools/uploader/documentation>

**Examples**

```

token = '420b4672ebfc43bab48dc0d18a32fb6f'
misc_upload_cli(auth_token = token,
                uploader = '~/sbg-uploader/',
                file = '~/example.fastq', project_id = '1234')

```

---

pipeline_add	<i>Add a pipeline to a specified project</i>
--------------	--

---

### Description

Add a pipeline to a specified project. You can use this function to add a pipeline from your other project or a public pipeline to a project.

### Usage

```
pipeline_add(auth_token = NULL, project_id_to = NULL,
             project_id_from = NULL, pipeline_id = NULL, revision = NULL, ...)
```

### Arguments

auth_token	auth token
project_id_to	ID of a project you to copy pipeline into.
project_id_from	ID of the project you wish to add from. Specify values such as "my" to specify a pipeline from "My Pipelines" section or omit for a public pipeline, respectively.
pipeline_id	ID of the pipeline you wish to add to project.
revision	Revision of the pipeline you wish to add to the project.
...	parameters passed to sbgapi function

### Value

parsed list of the returned json

### Examples

```
token = '58aeb140-1970-0130-6386-001f5b34aa78'
req = pipeline_add(token,
                   project_id_to = '7f7a72d0-da77-4f51-9416-99f14f7316ab',
                   project_id_from = 'f0eb447f-3511-4b28-9253-eba96191d432',
                   pipeline_id = '53452130d79f0049c0c94441')
```

---

pipeline_details	Returns the details of a pipeline for a project
------------------	---

---

### Description

Returns the details of a pipeline (runtime and regular parameters, description etc.) for a project.

### Usage

```
pipeline_details(auth_token = NULL, project_id = NULL, pipeline_id = NULL,  
...)
```

### Arguments

auth_token	auth token
project_id	ID of a project you want to access.
pipeline_id	ID of a pipeline you want to access.
...	parameters passed to sbgapi function

### Details

When using the API to run a task, the user needs to set input files for all input nodes. To facilitate this, some pipeline input nodes may contain field "suggested files", that contains files which may be used as default input (reference genomes, SNP database, etc.).

### Value

parsed list of the returned json

### Examples

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'  
req = pipeline_details(token,  
    project_id = 'b0b3a611-6bb0-47e5-add7-a83402cf7858',  
    pipeline_id = '55606ad4896a5d524656afd0')
```



---

pipeline\_list\_my      *Returns the list of pipelines in user's "My Pipelines" section*

---

**Description**

Returns the list of pipelines in user's "My Pipelines" section.

**Usage**

```
pipeline_list_my(auth_token = NULL, ...)
```

**Arguments**

auth_token	auth token
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'  
req = pipeline_list_my(token)
```

---

pipeline\_list\_project      *Returns a list of all the pipelines in project*

---

**Description**

Returns a list of all the pipelines in project.

**Usage**

```
pipeline_list_project(auth_token = NULL, project_id = NULL, ...)
```

**Arguments**

auth_token	auth token
project_id	ID of a project you want to access.
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'
req = pipeline_list_project(token,
    project_id = 'b0b3a611-6bb0-47e5-add7-a83402cf7858')
```

---

pipeline\_list\_pub      *Returns the list of all public pipelines*

---

**Description**

Returns the list of all public pipelines.

**Usage**

```
pipeline_list_pub(auth_token = NULL, ...)
```

**Arguments**

auth_token	auth token
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'
req = pipeline_list_pub(token)
```

---

project\_delete      *Delete a project*

---

**Description**

Note that this deletes all files, tasks which belong to a project.

**Usage**

```
project_delete(auth_token = NULL, project_id = NULL, ...)
```

**Arguments**

auth_token	auth token
project_id	ID of a project you want to delete.
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'  
req = project_delete(token,  
    project_id = '3a21ade8-ef3e-41f8-8ac2-1dc3b434ac77')
```

---

project_details	<i>Returns the details of the project</i>
-----------------	---

---

**Description**

Returns the details of the project.

**Usage**

```
project_details(auth_token = NULL, project_id = NULL, ...)
```

**Arguments**

- auth\_token      auth token
- project\_id      ID of a project you want to access.
- ...              parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'  
req = project_details(token,  
    project_id = 'b0b3a611-6bb0-47e5-add7-a83402cf7858')
```

---

project_list	Returns the list of all projects you have access to
--------------	---

---

**Description**

Returns the list of all projects you have access to.

**Usage**

```
project_list(auth_token = NULL, ...)
```

**Arguments**

auth_token	auth token
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'  
req = project_list(token)
```

---

project_members	Returns a list of all users invited to the project and their privileges
-----------------	---

---

**Description**

Returns a list of all users invited to the project and their privileges. Project ID is specified as path parameter. Call returns ID and username of the user with privileges.

**Usage**

```
project_members(auth_token = NULL, project_id = NULL, ...)
```

**Arguments**

auth_token	auth token
project_id	ID of a project you want to access.
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json



---

project\_member\_delete *Removes a member from a project*

---

### Description

Note that user\_id parameter is not username, but user ID parameter that you can receive from GET members call.

### Usage

```
project_member_delete(auth_token = NULL, project_id = NULL,
    user_id = NULL, ...)
```

### Arguments

auth_token	auth token
project_id	ID of a project you want to access.
user_id	ID of the user you wish to remove.
...	parameters passed to sbgapi function

### Value

parsed list of the returned json

### Examples

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'
req = project_member_delete(token,
    project_id = '7f7a72d0-da77-4f51-9416-99f14f7316ab',
    user_id = '08890148-6d9e-4a10-b284-924228d3f99a')
```

---

project\_member\_update *Set permissions for a user to a project*

---

### Description

This call will set project's member privileges. Privileges you do not explicitly set to "true" will be automatically set to "false". Project ID and user ID are specified in path parameters. Note that you must get the user IDs by performing the project\_members() call and gathering id of the user with a specific permission.

### Usage

```
project_member_update(auth_token = NULL, project_id = NULL,
    user_id = NULL, write = FALSE, copy = FALSE, execute = FALSE,
    admin = FALSE, ...)
```

**Arguments**

auth_token	auth token
project_id	ID of a project you want to access.
user_id	ID of a user whose permissions you wish to set
write	Logical. Ability to create/edit/delete project objects.
copy	Logical. Ability to download or copy files.
execute	Logical. Ability to run tasks.
admin	Logical. User has all rights on the project (including changing).
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '58aeb140-1970-0130-6386-001f5b34aa78'
req = project_member_update(token,
    project_id = '7f7a72d0-da77-4f51-9416-99f14f7316ab',
    user_id = '08890148-6d9e-4a10-b284-924228d3f99a')
```

---

project\_new

*Create new project*

---

**Description**

You can use this call to create a project. All details, including project name, description and funding source are specified as part of the JSON, sent as the body of the request. This call returns details of the project.

**Usage**

```
project_new(auth_token = NULL, name = NULL, description = NULL,
    billing_group_id = NULL, ...)
```

**Arguments**

auth_token	auth token
name	Name of the project you wish to create.
description	Description of the project you wish to create.
billing_group_id	ID of the billing group you wish to use for this project.
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'
req = project_new(token, name = 'Test API project',
                  description = 'My first API project',
                  billing_group_id = '5b6d5e71-dff8-42fc-8583-500d858f1093')
```

---

sbgapi

*wrapper of http logic for SBG API*


---

**Description**

wrapper of http logic for SBG API

**Usage**

```
sbgapi(auth_token = NULL, version = "1.1", path, method = c("GET", "POST",
"PUT", "DELETE"), query = NULL, body = list(),
base_url = paste0("https://api.sbgenomics.com/", version, "/"))
```

**Arguments**

auth_token	authenticate token string.
version	API version number, default 1.1.
path	path connected with base_url.
method	one of 'GET', 'POST', 'PUT', 'Delete'
query	Passed to httr package GET/POST call.
body	Passed to httr package GET/POST/PUT/DELETE call.
base_url	default is 'https://api.sbgenomics.com/1.1'

**Details**

Used for advanced users and the core method for higher level API in this package, please refer to the easy api manual and the two vignettes pages for more convenient usage.

**Value**

returned request list of httr

**References**

<https://docs.sbgenomics.com/display/developerhub/API>



**Examples**

```
token <- "fake_token"

## list projects
sbgapi(auth_token = token, path = 'project', method = "GET")
```

---

Task-class	<i>Task class</i>
------------	-------------------

---

**Description**

Task class

**Details**

A task execution require auth, project\_id, pipeline\_id and inputs parameters, there are two ways to execute a task, the recommended way is to use a cascading method to create a project object called p then just call p\$task\_run() to pass your parameters. This way you save your time passing auth and project\_id. The other way is to create a Task object with all required fields and call run method. Please check example in the end or tutorial for easy API.

**Fields**

id [characterORNULL] The task ID number, used when referring to the task in other Seven Bridges API calls

name [characterORNULL] Name of the task you wish to execute. If this is not specified, the task will be named automatically.

description [characterORNULL] Description of the task you wish to execute.

pipeline\_id [characterORNULL] ID of the pipeline you wish to execute.

pipeline\_revision [characterORNULL] Revision number of the pipeline you wish to execute. If this is not specified, the latest pipeline revision is used.

start\_time [numericORNULL] start time.

status [characterORNULL] 1) active: task is currently running. 2) completed: task has finished successfully. 3) aborted: task was aborted by user. 4) failed: task has failed to finish due to either bad inputs and/or parameters, or because of the internal infrastructure failures.

message [characterORNULL] task message

jobs\_completed [numericORNULL] completed jobs

jobs\_total [numericORNULL] total jobs.

inputs [listORNULL] required for task execution. List of key-value pairs containing mappings of pipeline input node ID to file IDs. Note that you must supply an array of file IDs for each input nodes, even if the array is empty.

parameters [listORNULL] required for task execution. List of key-value pairs containing mappings of node IDs to apps specific parameters. Note that you must supply some value for parameters, even if this an empty list of key-value pairs.

project\_id [characterORNULL] required for task execution. ID of the project you want to execute the task in.

## Examples

```
token <- "aef7e9e3f6c54fb1b338ac4ecddf1a56"
a <- Auth(token)
## A task constructor
Task(auth = Auth(token),
      name = "my task",
      description = "A text description",
      pipeline_id = "fake_pipeline_id",
      project_id = "fake_project_id",
      inputs = list(
        "177252" = list("fake_id")
      ))

## replace with real token then follow the examples here
## get billing info
b <- a$billing()
p <- a$project("API")
## get the pipeline from your project not public one
f.pipe <- p$pipeline(name = "FastQC")
## check the inputs needed for running tasks
f.pipe$details()
## Ready to run a task? go
f.task <- p$task_run(name = "my task",
                    description = "A text description",
                    pipeline_id = f.pipe$id,
                    inputs = list(
                      "177252" = list(f.file$id)
                    ))
f.task$run()
## or you can just run with Task constructor
f.task <- Task(auth = Auth(token),
              name = "my task",
              description = "A text description",
              pipeline_id = f.pipe$id,
              project_id = p$id,
              inputs = list(
                "177252" = list(f.file$id)
              ))
## Monitor you task
f.task$monitor(30)

## download a task output files
f.task <- p$task("my task")
f.task$download("~/Desktop/")
```

```
## Abort the task
f.task$abort()
```

---

task_action	<i>Performs action on the task</i>
-------------	------------------------------------

---

### Description

Performs action on the task.

### Usage

```
task_action(auth_token = NULL, project_id = NULL, task_id = NULL,
  action = "abort", ...)
```

### Arguments

auth_token	auth token
project_id	ID of a project you want to access.
task_id	ID of a task you want to access.
action	Character string specifying the action. Currently, only supported action is 'abort'.
...	parameters passed to sbgapi function

### Value

parsed list of the returned json

### Examples

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'
req = task_action(token,
  project_id = '7f7a72d0-da77-4f51-9416-99f14f7316ab',
  task_id = '5506a44ae4b04a4ab3ae7250',
  action = 'abort')
```

---

task_details	Returns information about the task
--------------	------------------------------------

---

### Description

Returns information about the task.

### Usage

```
task_details(auth_token = NULL, project_id = NULL, task_id = NULL,  
            download.url = FALSE, ...)
```

### Arguments

auth_token	auth token
project_id	ID of a project you want to access.
task_id	ID of a task you want to access.
download.url	Logical. Return the download URL or not.
...	parameters passed to sbgapi function

### Details

Each task has a status and status message, containing the more detailed information about the task status, associated with it. This is a list of all values that task status can have:

- active - Task is currently running.
- completed - Task has finished successfully.
- aborted - Task was aborted by user.
- failed - Task has failed to finish due to either bad inputs and/or parameters, or because of the internal infrastructure failures.

### Value

parsed list of the returned json

### Examples

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'  
req1 = task_details(token,  
                   project_id = '1c1d06d2-5862-48f6-b595-e0099b20937e',  
                   task_id = '22237')  
req2 = task_details(token,  
                   project_id = '1c1d06d2-5862-48f6-b595-e0099b20937e',  
                   task_id = '22237', download.url = TRUE)
```

---

task_list	<i>Returns the list of all the tasks for a project</i>
-----------	--

---

**Description**

Returns the list of all the tasks for a project.

**Usage**

```
task_list(auth_token = NULL, project_id = NULL, ...)
```

**Arguments**

auth_token	auth token
project_id	ID of a project you want to access.
...	parameters passed to sbgapi function

**Details**

This function returns general information and status of a task, in case you want to get a details, including the inputs, outputs and parameters set for that task, you will have to use task details resource referencing the task\_id of a task that you want to get information about.

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'  
req = task_list(token,  
                '1c1d06d2-5862-48f6-b595-e0099b20937e')
```

---

task_run	<i>Runs a task as a part of a project</i>
----------	---

---

**Description**

Runs a task as a part of a project.

**Usage**

```
task_run(auth_token = NULL, project_id = NULL, task_details = NULL, ...)
```

**Arguments**

auth_token	auth token
project_id	ID of a project you want to access.
task_details	A list with the following components: <ul style="list-style-type: none"> <li>• pipeline_id - ID of the pipeline you wish to execute</li> <li>• pipeline_revision - Revision of the pipeline you wish to execute. If not specified, latest revision is used.</li> <li>• name - Name of the task you wish to execute</li> <li>• description - Description of the task you wish to execute</li> <li>• inputs - Named list containing mappings of pipeline input node ID to file IDs. Note that file IDs always need to be specified as an list, even if empty or with one element.</li> <li>• parameters - Named list containing mappings of node IDs to apps specific parameters. Note that parameters are always specified as an list, even if empty or with one element.</li> </ul>
...	parameters passed to sbgapi function

**Details**

All the details, including the pipeline ID and runtime parameters, are specified via a list. See the example for details.

**Value**

parsed list of the returned json

**Examples**

```
token = '58aeb140-1970-0130-6386-001f5b34aa78'
details = list(
    'name' = 'Test 2 of C. Elegans VC',
    'description' = 'Testing Caenorhabditis elegans Exome Variant Calling',
    'pipeline_id' = '422',
    'inputs' = list('309485' = 13645,
                   '317344' = 13646,
                   '318662' = 13645,
                   '699018' = 13647),
    'parameters' = list('393463' = list('read_trimming_qual' = 30,
                                       'rg_seq_tech' = 'Illumina'),
                       '677492' = list()))

req = task_run(token,
               project_id = '7f7a72d0-da77-4f51-9416-99f14f7316ab',
               task_details = details)
```

---

upload\_complete\_all    *Reports the complete file upload*

---

**Description**

If the whole parts are uploaded, and the provided ETags are correct, then the file is assembled and made available on the SBG platform.

**Usage**

```
upload_complete_all(auth_token = NULL, upload_id = NULL, ...)
```

**Arguments**

auth_token	auth token
upload_id	ID of the upload
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '58aeb140-1970-0130-6386-001f5b34aa78'  
req = upload_complete_all(token,  
    upload_id = '8D7sQJxQk14ubsEnKaoeQZ1RvV6ouQtMzBwaQNJdxPDLypUC3WogwtJdncevHxnT')
```

---

upload\_complete\_part    *Reports the completion of the part upload*

---

**Description**

The ETag is provided for the correctness check upon completion of the whole upload. Value for the ETag is provided by AWS S3 service when uploading the file in the ETag header.

**Usage**

```
upload_complete_part(auth_token = NULL, upload_id = NULL,  
    part_number = NULL, e_tag = NULL, ...)
```

**Arguments**

auth_token	auth token
upload_id	ID of the upload
part_number	ID of the part you wish to report as completed
e_tag	Value of the ETag header returned by AWS S3 when uploading part of the file.
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '58aeb140-1970-0130-6386-001f5b34aa78'
req = upload_complete_part(token,
    upload_id = '8D7sQJxQk14ubsEnKaoeQZlRvV6ouQtMzBwaQNJdxPDLyUC3WogwtJdncevHxnT',
    part_number = '1',
    e_tag = 'd41d8cd98f00b204e9800998ecf8427e')
```

---

upload_delete	<i>Aborts the upload</i>
---------------	--------------------------

---

**Description**

All upload records and the file are deleted.

**Usage**

```
upload_delete(auth_token = NULL, upload_id = NULL, ...)
```

**Arguments**

auth_token	auth token
upload_id	ID of the upload
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'
req = upload_delete(token,
    upload_id = '8D7sQJxQk14ubsEnKaoeQZlRvV6ouQtMzBwaQNJdxPDLyUC3WogwtJdncevHxnT')
```



---

upload_info	<i>Returns upload information for the ongoing upload</i>
-------------	--

---

**Description**

Returns the upload information for the ongoing upload.

**Usage**

```
upload_info(auth_token = NULL, upload_id = NULL, ...)
```

**Arguments**

auth_token	auth token
upload_id	ID of the upload
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'
req = upload_info(token,
                  upload_id = '8D7sQJxQk14ubsEnKaoeQZ1RvV6ouQtMzBWaQNJdxPDLypUC3WogwtJdncevHxnT')
```

---

upload_info_part	<i>Returns AWS S3 signed URL for a part of the file upload</i>
------------------	--

---

**Description**

Gets the signed URL for the upload of the specified part. Note that URLs are valid for 60 seconds only and that you should initiate upload to the signed URL in this time frame.

**Usage**

```
upload_info_part(auth_token = NULL, upload_id = NULL, part_number = NULL,
                 ...)
```

**Arguments**

auth_token	auth token
upload_id	ID of the upload
part_number	Number of the upload file part that you wish to access
...	parameters passed to sbgapi function

**Value**

parsed list of the returned json

**Examples**

```
token = '420b4672ebfc43bab48dc0d18a32fb6f'
req = upload_info_part(token,
                        upload_id = 'aVluXRqSX2bse6va3AFFgVApp0CQ9IABeA8HnyyiEw85j6pNyV989H4xvJpr53xa',
                        part_number = 1)
```

---

upload\_init

*Initializes the upload of the specified file*

---

**Description**

This is the first operation performed when you wish to upload a file. Operation is initialized by providing file name, project id where you wish the file to be uploaded to (if not specified, defaults to user's stash) and optionally by providing wanted part size. You may wish to set your part size to a low value if you experience problems with uploading large file parts, although default value of 5MB should be good enough for most users.

**Usage**

```
upload_init(auth_token = NULL, project_id = NULL, name = NULL,
            size = NULL, part_size = NULL, ...)
```

**Arguments**

auth_token	auth token
project_id	ID of the project you wish to upload to
name	Name of the file you wish to upload
size	Size of the file you wish to upload
part_size	Requested part size. Note that API may reject your requested part size and return proper one in response.
...	parameters passed to sbgapi function

**Details**

Limits:

- Maximum number of parts is 10000
- Maximum file size is 5TB
- Maximum part size is 5GB
- Default part size is 5MB

**Value**

parsed list of the returned json

**Examples**

```
token = '58aeb140-1970-0130-6386-001f5b34aa78'  
req = upload_init(token,  
                  project_id = 'f0eb447f-3511-4b28-9253-eba96191d432',  
                  name = 'Sample1_RNASeq_chr20.pe_1.fastq', size = 5242880)
```

# Index

Auth (Auth-class), 3  
Auth-class, 3

billing, 4

file\_copy, 6  
file\_delete, 7  
file\_details, 7  
file\_download\_url, 8  
file\_list, 9  
file\_meta\_update, 9  
file\_type (FileTypeSingleEnum-class), 5  
FileTypeSingleEnum-class, 5

Item (Item-class), 11  
Item-class, 11

Metadata (FileTypeSingleEnum-class), 5  
Metadata-class  
    (FileTypeSingleEnum-class), 5

misc\_get\_auth\_token, 11  
misc\_get\_uploader, 12  
misc\_make\_metadata, 12  
misc\_upload\_cli, 14

paired\_end (FileTypeSingleEnum-class), 5  
pipeline\_add, 15  
pipeline\_details, 16  
pipeline\_list\_my, 17  
pipeline\_list\_project, 17  
pipeline\_list\_pub, 18  
project\_delete, 18  
project\_details, 19  
project\_list, 20  
project\_member\_add, 21  
project\_member\_delete, 22  
project\_member\_update, 22  
project\_members, 20  
project\_new, 23

qual\_scale (FileTypeSingleEnum-class), 5

sbgapi, 24  
sbgr-package, 2  
seq\_tech (FileTypeSingleEnum-class), 5

Task (Task-class), 25  
Task-class, 25  
task\_action, 27  
task\_details, 28  
task\_list, 29  
task\_run, 29

upload\_complete\_all, 31  
upload\_complete\_part, 31  
upload\_delete, 32  
upload\_info, 33  
upload\_info\_part, 33  
upload\_init, 34