BaseSpacePy Documentation

Release 0.2

Illumina

March 12, 2014

CONTENTS

Contents:

CHAPTER

ONE

AVAILABLE MODULES

1.1 API

class BaseSpacePy.api.BaseSpaceAPI.**BaseSpaceAPI**(*clientKey=None*, *clientSecret=None*, *apiServer=None*, *version=None*, *appSessionId=''*, *AccessToken=''*, *timeout=10*, *profile='DEFAULT'*)

The main API class used for all communication with the REST server

appResultFileUpload(*Id*, *localPath*, *fileName*, *directory*, *contentType*)

Uploads a file associated with an AppResult to BaseSpace and returns the corresponding file object. Small files are uploaded with a single-part upload method, while larger files (< 25 MB) are uploaded with multi-part upload.

Parameters

- **Id** – AppResult id.
- **localPath** – The local path to the file to be uploaded, including file name.
- **fileName** – The desired filename in the AppResult folder on the BaseSpace server.
- **directory** – The directory the file should be placed in on the BaseSpace server.
- **contentType** – The content-type of the file, eg. 'text/plain' for text files, 'application/octet-stream' for binary files

Returns a newly created File instance

createAppResult(*Id*, *name*, *desc*, *samples=None*, *appSessionId=None*)

Create an AppResult object.

Parameters

- **Id** – The id of the project in which the AppResult is to be added
- **name** – The name of the AppResult
- **desc** – A description of the AppResult
- **samples** – (Optional) A list of one or more Samples Ids that the AppResult is related to
- **appSessionId** – (Optional) If no appSessionId is given, the id used to initialize the BaseSpaceAPI instance will be used. If appSessionId is set equal to an empty string, a new appsession will be created for the appresult object

Raises Exception when attempting to create AppResult in an AppSession that has a status other than 'running'.

Returns a newly created AppResult instance

createProject(*Name*)
Creates a project with the specified name and returns a project object. If a project with this name already exists, the existing project is returned.

Parameters **Name** – Name of the project

Returns a Project instance of the newly created project

fileDownload(*Id*, *localDir*, *byteRange=None*, *createBsDir=False*)
Downloads a BaseSpace file to a local directory, and names the file with the BaseSpace file name. If the File has a directory in BaseSpace, it will be re-created locally in the provided localDir (to disable this, set createBsPath=False).

If the file is large, multi-part download will be used.

Byte-range requests are supported for only small byte ranges (single-part downloads). Byte-range requests are restricted to a single request of 'start' and 'end' bytes, without support for negative or empty values for 'start' or 'end'.

Parameters

- **Id** – The file id
- **localDir** – The local directory to place the file in
- **byteRange** – (optional) The byte range of the file to retrieve, provide a 2-element list with start and end byte values
- **createBsDir** – (optional) create BaseSpace File's directory inside localDir (default: False)

Raises ByteRangeException if the provided byte range is invalid

Returns a File instance

fileS3metadata(*Id*)
Returns the S3 url and etag (md5 for small files uploaded as a single part) for a BaseSpace file

Parameters **Id** – The file id

Raises Exception if REST API call to BaseSpace server fails

Returns Dict with s3 url ('url' key) and etag ('etag' key)

fileUrl(*Id*)
** Deprecated in favor of fileS3metadata() **

Returns URL of file (on S3)

Parameters **Id** – The file id

Raises Exception if REST API call to BaseSpace server fails

Returns a URL

filterVariantSet(*Id*, *Chrom*, *StartPos*, *EndPos*, *Format='json'*, *queryPars=None*)
List the variants in a set of variants. Note the maximum returned records is 1000.

Parameters

- **Id** – The id of the variant file
- **Chrom** – Chromosome name
- **StartPos** – The start position of the sequence of interest
- **EndPos** – The start position of the sequence of interest

- **Format** – (optional) Format for results, possible values: 'vcf' (not implemented yet), 'json'(default, which actually returns an object)
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a list of Variant instances, when Format is json; a string, when Format is vcf

getAccess(*obj*, *accessType='write'*, *web=False*, *redirectURL=''*, *state=''*)

Requests access to the provided BaseSpace object.

Parameters

- **obj** – The data object we wish to get access to – must be a Project, Sample, AppResult, or Run.
- **accessType** – (Optional) the type of access (browse|read|write|create), default is write. Create is only supported for Projects.
- **web** – (Optional) true if the App is web-based, default is false meaning a device based app
- **redirectURL** – (Optional) Redirect URL for the web-based case
- **state** – (Optional) A parameter that will passed through to the redirect response.

Raises ModelNotSupportedException for classes of objects not supported by this method

Returns for device requests, a dictionary of server response; for web requests, a url to to send the user to

getAccessibleRunsByUser(*queryPars=None*)

Returns a list of accessible runs for the current User

Parameters **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a list of Run instances

getAppResultById(*Id*, *queryPars=None*)

Returns an AppResult object corresponding to Id

Parameters

- **Id** – The Id of the AppResult
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns an AppResult instance

getAppResultFiles(*Id*, *queryPars=None*)

•Deprecated in favor of getAppResultFileById() *

Returns a list of File object for an AppResult

Parameters

- **Id** – The id of the AppResult
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a list of File instances

getAppResultFilesById(*Id*, *queryPars=None*)

Returns a list of File object for an AppResult

Parameters

- **Id** – The id of the AppResult

- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a list of File instances

getAppResultPropertiesById(*Id*, *queryPars=None*)
Returns the Properties of an AppResult object corresponding to AppResult Id

Parameters

- **Id** – The Id of the AppResult
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a PropertyList instance

getAppResultsByProject(*Id*, *queryPars=None*, *statuses=None*)
Returns a list of AppResult object associated with the project with Id

Parameters

- **Id** – The project id
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering
- **statuses** – An (optional) list of AppResult statuses to filter by, eg., 'complete'

Returns a list of AppResult instances

getAppSession(*Id=None*)
Get metadata about an AppSession. Note that the client key and secret must match those of the AppSession's Application.

Parameters **Id** – an AppSession Id; if not provided, the AppSession Id of the BaseSpaceAPI instance will be used

Returns An AppSession instance

getAppSessionById(*Id*)
Get metadata about an AppSession. Note that the client key and secret must match those of the AppSession's Application.

Parameters **Id** – The Id of the AppSession

Returns An AppSession instance

getAppSessionInputsById(*Id*, *queryPars=None*)
Returns the input properties of an AppSession

Parameters

- **Id** – An AppSessionId
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a dictionary of input properties, keyed by input Name

getAppSessionPropertiesById(*Id*, *queryPars=None*)
Returns the Properties of an AppSession

Parameters

- **Id** – An AppSession Id
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns A PropertyList instance

getAppSessionPropertyByName(*Id*, *name*, *queryPars=None*)
Returns the multi-value Property of the provided AppSession that has the provided Property name. Note - this method (and REST API) is supported for ONLY multi-value Properties.

Parameters

- **Id** – The AppSessionId
- **name** – Name of the multi-value property to retrieve
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns A multi-value propertylist instance such as MultiValuePropertyAppResultsList (depending on the Property Type)

getAvailableGenomes(*queryPars=None*)
Returns a list of all available genomes

Parameters **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a list of GenomeV1 instances

getCoverageMetaInfo(*Id*, *Chrom*)
Returns metadata about coverage of a chromosome. Note that HrefCoverage must be available for the provided BAM file

Parameters

- **Id** – the Id of a Bam file
- **Chrom** – chromosome name

Returns a CoverageMetaData instance

getFileById(*Id*, *queryPars=None*)
Returns a file object by Id

Parameters

- **Id** – The id of the file
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a File instance

getFilePropertiesById(*Id*, *queryPars=None*)
Returns the Properties of a file object by Id

Parameters

- **Id** – The id of the file
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a PropertyList instance

getFilesBySample(*Id*, *queryPars=None*)

•Deprecated in favor of getSampleFilesById() *

Returns a list of File objects associated with a Sample

Parameters

- **Id** – A Sample id
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a list of File instances

getGenomeById(*Id*)
Returns an instance of Genome with the specified Id

Parameters **Id** – The genome id

Returns a GenomeV1 instance

getIntervalCoverage(*Id*, *Chrom*, *StartPos*, *EndPos*)
Returns metadata about an alignment, including max coverage and cov granularity. Note that HrefCoverage must be available for the provided BAM file.

Parameters

- **Id** – the Id of a BAM file
- **Chrom** – chromosome name
- **StartPos** – get coverage starting at this position
- **EndPos** – get coverage up to and including this position; the returned EndPos may be larger than requested due to rounding up to nearest window end coordinate

Returns a Coverage instance

getProjectById(*Id*, *queryPars=None*)
Request a project object by Id

Parameters

- **Id** – The Id of the project
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a Project instance

getProjectByUser(*queryPars=None*)
Returns a list available projects for the current User.

Parameters **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a list of Project instances

getProjectPropertiesById(*Id*, *queryPars=None*)
Request the Properties of a project object by Id

Parameters

- **Id** – The Id of the project
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a ProjectList instance

getRunById(*Id*, *queryPars=None*)
Request a run object by Id

Parameters

- **Id** – The Id of the run
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a Run instance

getRunFilesById(*Id*, *queryPars=None*)
Request the files associated with a Run, using the Run's Id

Parameters

- **Id** – The Id of the run
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a list of Run instances

getRunPropertiesById(*Id*, *queryPars=None*)
Request the Properties of a run object by Id

Parameters

- **Id** – The Id of the run
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a PropertyList instance

getRunSamplesById(*Id*, *queryPars=None*)
Request the Samples associated with a Run, using the Run's Id

Parameters

- **Id** – The Id of the run
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a list of Sample instances

getSampleById(*Id*, *queryPars=None*)
Returns a Sample object

Parameters

- **Id** – The id of the sample
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a Sample instance

getSampleFilesById(*Id*, *queryPars=None*)
Returns a list of File objects associated with a Sample

Parameters

- **Id** – A Sample id
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a list of File instances

getSamplePropertiesById(*Id*, *queryPars=None*)
Returns the Properties of a Sample object

Parameters

- **Id** – The id of the sample
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a PropertyList instance

getSamplesByProject(*Id*, *queryPars=None*)
Returns a list of samples associated with a project with Id

Parameters

- **Id** – The id of the project
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a list of Sample instances

getUserById(*Id*)
Returns the User object corresponding to User Id

Parameters **Id** – The Id of the user

Returns a User instance

getVariantMetadata(*Id*, *Format='json'*)
Returns the header information of a VCF file.

Parameters

- **Id** – The Id of the VCF file
- **Format** – (optional) The return-value format, set to 'json' (default) to return return an object (not actually json format), or 'vcf' (not implemented yet) to return a string in VCF format.

Returns A VariantHeader instance

getVerificationCode(*scope*)
For non-web applications (eg. devices), returns the device code and verification url for the user to approve access to a specific data scope.

Parameters **scope** – The scope that access is requested for (e.g. 'browse project 123')

Returns dictionary of server response

getWebVerificationCode(*scope*, *redirectURL*, *state=''*)
Generates the URL the user should be redirected to for web-based authentication

Parameters

- **scope** – The scope that access is requested for (e.g. 'browse project 123')
- **redirectURL** – The redirect URL
- **state** – (Optional) A state parameter that will passed through to the redirect response

Returns a url

multipartFileDownload(*Id*, *localDir*, *processCount=10*, *partSize=25*, *createBsDir=False*, *tempDir=''*)
Method for multi-threaded file-download for parallel transfer of very large files (currently only runs on unix systems)

Parameters

- **Id** – The ID of the File to download
- **localDir** – The local path in which to store the downloaded file
- **processCount** – (optional) The number of processes to be used, default 10
- **partSize** – (optional) The size in MB of individual file parts to download, default 25
- **createBsDir** – (optional) create BaseSpace File's directory in local_dir, default False
- **tempDir** – (optional) Set temp directory to use debug mode, which stores downloaded file chunks in individual files, then completes by 'cat'ing chunks into large file

Returns a File instance

multipartFileUpload(*Id*, *localPath*, *fileName*, *directory*, *contentType*, *tempDir=None*, *processCount=10*, *partSize=25*)

Method for multi-threaded file-upload for parallel transfer of very large files (currently only runs on unix systems)

Parameters

- **Id** – The AppResult ID
- **localPath** – The local path of the file to upload, including file name; local path will not be stored in BaseSpace (use directory argument for this)
- **fileName** – The desired filename on the server
- **directory** – The desired directory name on the server (empty string will place it in the root directory)
- **contentType** – The content type of the file
- **tempdir** – (optional) Temp directory to use for temporary file chunks to upload
- **processCount** – (optional) The number of processes to be used, default 10
- **partSize** – (optional) The size in MB of individual upload parts (must be >5 Mb and <=25 Mb), default 25

Returns a File instance, which has been updated after the upload has completed.

obtainAccessToken(*code*, *grantType='device'*, *redirect_uri=None*)

Returns a user specific access token, for either device (non-web) or web apps.

Parameters

- **code** – The device code returned by the getVerificationCode method
- **grantType** – Grant-type may be either 'device' for non-web apps (default), or 'authorization_code' for web apps
- **redirect_uri** – The uri to redirect to; required for web apps only.

Raises OAuthException when redirect_uri isn't provided by web apps

Returns an access token

setAppSessionState(*Id*, *Status*, *Summary*)

Set the Status and StatusSummary of an AppSession in BaseSpace. Note - once Status is set to Completed or Aborted, no further changes can made.

Parameters

- **Id** – The id of the AppSession
- **Status** – The AppSession status string, must be one of: Running, Complete, NeedsAttention, TimedOut, Aborted
- **Summary** – The status summary string

Returns An updated AppSession instance

updatePrivileges(*code*, *grantType='device'*, *redirect_uri=None*)

Retrieves a user-specific access token, and sets the token on the current object.

Parameters

- **code** – The device code returned by the getVerificationCode method

- **grantType** – Grant-type may be either 'device' for non-web apps (default), or 'authorization_code' for web apps
- **redirect_uri** – The uri to redirect to; required for web apps only.

Returns None

1.2 Models

1.2.1 Project

class BaseSpacePy.model.Project.**Project**

Represents a BaseSpace Project object.

createAppResult(*api*, *name*, *desc*, *samples=None*, *appSessionId=None*)

Return a newly created app result object

Parameters

- **api** – An instance of BaseSpaceAPI
- **name** – The name of the app result
- **desc** – A description of the app result
- **samples** – (Optional) A list of one or more Samples Ids that the AppResult is related to
- **appSessionId** – (Optional) If no appSessionId is given, the id used to initialize the BaseSpaceAPI instance will be used. If appSessionId is set equal to an empty string, a new appsession will be created for the appresult object

getAccessStr(*scope='write'*)

Returns the scope-string to used for requesting BaseSpace access to the object

Parameters **scope** – The scope-type that is request (write|read)

getAppResults(*api*, *queryPars=None*, *statuses=None*)

Returns a list of AppResult objects.

Parameters

- **api** – An instance of BaseSpaceAPI
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering
- **statuses** – An optional list of statuses, eg. 'complete'

getSamples(*api*, *queryPars=None*)

Returns a list of Sample objects.

Parameters

- **api** – An instance of BaseSpaceAPI
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

isInit()

Tests if the Project instance has been initialized.

Raises ModelNotInitializedException if the Id variable is not set

Returns True on success

1.2.2 AppSession and AppResult

class BaseSpacePy.model.AppSession.**AppSession**
Returned from getAppSessionById() and getAppSesssion()

class BaseSpacePy.model.AppResult.**AppResult**

getAccessStr(*scope='write'*)
Returns the scope-string to be used for requesting BaseSpace access to the object

Parameters **scope** – The scope-type that is request (write|read)

getFiles(*api*, *queryPars=None*)
Returns a list of file objects

Parameters

- **api** – An instance of BaseSpaceAPI
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

getReferencedSamples(*api*)
Returns the sample objects for the referenced sample(s). NOTE this method makes one request to REST server per sample. If other reference types than Samples are present (they shouldn't be), they are ignored.

Parameters **api** – A BaseSpaceAPI instance

Returns A list of sample objects that are referenced by the AppResult.

getReferencedSamplesIds()
Return the sample ids for the referenced sample(s). If other reference types than Samples are present (they shouldn't be), they are ignored.

Returns a list of sample ids for the referenced samples.

isInit()
Tests if the AppResult instance has been initialized.

Raises ModelNotInitializedException if the Id variable is not set

Returns True on success

uploadFile(*api*, *localPath*, *fileName*, *directory*, *contentType*)
Uploads a local file to the BaseSpace AppResult

Parameters

- **api** – An instance of BaseSpaceAPI
- **localPath** – The local path of the file (including file name)
- **fileName** – The file name to use on the server
- **directory** – The remote directory to upload the file to on the server
- **contentType** – The content-type of the file

1.2.3 Sample

class BaseSpacePy.model.Sample.**Sample**
A BaseSpace Sample object.

getAccessStr(*scope='write'*)
Returns the scope-string to used for requesting BaseSpace access to the sample.

Parameters **scope** – The scope type that is request (eg. write, read).

getFiles(*api*, *queryPars=None*)
Returns a list of File objects

Parameters

- **api** – A BaseSpaceAPI instance
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

getReferencedAppResults(*api*)
Return the AppResults referenced by this sample. Note the returned AppResult objects do not have their "References" field set, to get a fully populated AppResult object you must use getAppResultById in BaseSpaceAPI. If other reference types than AppResults are present (they shouldn't be), they are ignored.

Parameters **api** – A BaseSpaceAPI instance

Returns A list of AppResult that are referenced by this sample (with their References field not populated)

isInit()
Tests if the Sample instance has been initialized.

Raises ModelNotInitializedException if the Id variable is not set

Returns True on success

1.2.4 File

class BaseSpacePy.model.File.**File**
Represents a BaseSpace file object

downloadFile(*api*, *localDir*, *byteRange=None*, *createBsDir=False*)
Download the file object to the specified localDir or a byte range of the file, by specifying the start and stop byte in the range.

Parameters

- **api** – A BaseSpaceAPI with read access on the scope including the file object.
- **localDir** – The local directory to place the file in.
- **byteRange** – (optional) Specify the start and stop byte of the file chunk that needs retrieved (as a 2-element list).
- **createBsDir** – (optional) create BaseSpace File's directory inside localDir (default: False)

filterVariant(*api*, *Chrom*, *StartPos*, *EndPos*, *Format='json'*, *queryPars=None*)
List the variants in a set of variants. Note the maximum returned records is 1000.

Parameters

- **api** – An instance of BaseSpaceAPI
- **Chrom** – Chromosome name
- **StartPos** – The start position of region of interest as a string
- **EndPos** – The end position of region of interest as a string

- **Format** – (optional) Format for results, possible values: 'vcf' (not implemented yet), 'json'(default, which actually returns an object)
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

Returns a list of Variant objects, when Format is json; a string, when Format is vcf

getCoverageMeta(*api*, *Chrom*)
Returns metadata about an alignment, including max coverage and cov granularity. Note that HrefCoverage must be available for the provided BAM file.

Parameters

- **api** – An instance of BaseSpaceAPI.
- **Chrom** – Chromosome name

Returns a CoverageMetaData object

getFileS3metadata(*api*)
Returns the S3 url and etag (md5 for small files uploaded as a single part) for a BaseSpace file

Parameters **api** – A BaseSpaceAPI with read access on the scope including the file object.

Returns Dict with s3 url ('url' key) and etag ('etag' key)

getFileUrl(*api*)
** Deprecated in favor of getFileS3metadata() **

Return the S3 url of the file.

Parameters **api** – A BaseSpaceAPI with read access on the scope including the file object.

getIntervalCoverage(*api*, *Chrom*, *StartPos*, *EndPos*)
Returns mean coverage levels over a sequence interval. Note that HrefCoverage must be available for the provided BAM file.

Parameters

- **api** – An instance of BaseSpaceAPI
- **Chrom** – Chromosome name as a string - for example 'chr2'
- **StartPos** – get coverage starting at this position
- **EndPos** – get coverage up to and including this position; the returned EndPos may be larger than requested due to rounding up to nearest window end coordinate

Returns A Coverage object

getVariantMeta(*api*, *Format='json'*)
Returns the header information of a VCF file.

Parameters

- **api** – An instance of BaseSpaceAPI
- **Format** – (optional) The return-value format, set to 'vcf' to VCF format (string) or 'json' (default, which acutally returns on object)

Returns A VariantHeader object

isInit()
Tests if the File instance has been initialized.

Raises ModelNotInitializedException if the Id variable is not set

Returns True on success

isValidFileOption(*filetype*)

** **Deprecated - HrefCoverage should be present for all BAM files in BaseSpace.** However, the attribute may be missing when there has been an error when BaseSpace internally generates coverage data from the BAM file. This is the same situation for HrefVariants on all VCF files. **

Is called to test if the File instance matches the filetype parameter

Parameters **filetype** – The filetype for coverage or variant requests (eg., 'bam', 'vcf')

1.2.5 Run

class BaseSpacePy.model.Run.**Run**

A BaseSpace Run object

getAccessStr(*scope='write'*)

Returns the scope-string to used for requesting BaseSpace access to the Run.

Parameters **scope** – The scope type that is request (eg. write, read).

getFiles(*api*, *queryPars=None*)

Returns a list of File objects associated with the Run

Parameters

- **api** – An instance of BaseSpaceAPI
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

getSamples(*api*, *queryPars=None*)

Returns a list of Sample objects associated with the Run

Parameters

- **api** – An instance of BaseSpaceAPI
- **queryPars** – An (optional) object of type QueryParameters for custom sorting and filtering

isInit()

Tests if the Run instance has been initialized.

Raises **ModelNotInitializedException** if the Id variable is not set

Returns True on success

1.2.6 QueryParameters

class BaseSpacePy.model.QueryParameters.**QueryParameters**(*pars=None*, *required=None*)

The QueryParameters class can be passed as an optional argument for sorting/filtering of list-responses (such as lists of samples, AppResults, variants, etc.)

validate()

Validates that query parameter keys and values are properly formed: required keys are present, and keys and values are within the set of known acceptable keys/values.

Raises

- **UndefinedParameterException** – when a required parameter is not present
- **UnknownParameterException** – when a parameter name is not present in the list of acceptable parameters names

- **IllegalParameterException** – when a parameter value (with a valid name) is not present in the list of acceptable parameters values

Returns None

CHAPTER

TWO

GETTING STARTED

2.1 Introduction

`BaseSpacePy` is a Python based SDK to be used in the development of Apps and scripts for working with Illumina's BaseSpace cloud-computing solution for next-gen sequencing data analysis. The primary purpose of the SDK is to provide an easy-to-use Python environment enabling developers to authenticate a user, retrieve data, and upload data/results from their own analysis to BaseSpace.

If you haven't already done so, you may wish to familiarize yourself with the BaseSpace developer documentation (https://developer.basespace.illumina.com/) prior to working through the example scripts below.

Note: It will be necessary to have created a BaseSpace account with a new App and have the `client_key` and `client_secret` codes for the App available to run a number of the following examples.

2.1.1 Availability

Version 0.1 of `BaseSpacePy` can be checked out here:

```
git clone git@github.com:basespace/basespace-python-sdk.git
```

2.1.2 Setup

Requirements: Python 2.6 with the packages 'urllib2', 'pycurl', 'multiprocessing' and 'shutil' available.

The multi-part file upload will currently only run on a unix setup.

To install 'BaseSpacePy' run the 'setup.py' script in the `src` directory (for a global install you will need to run this command with root privileges):

```
cd basespace-python-sdk/src
python setup.py install
```

If you do not have root access, you may use the `--prefix` option to specify the install directory (make sure this directory is in you PYTHONPATH):

```
python setup.py install --prefix=/folder/in/my/pythonpath
```

For more install options type:

```
python setup.py --help
```

Altenatively you may include the src directory in your PYTHONPATH by doing the following export:

```
export PYTHONPATH=$PYTHONPATH:/my/path/basespace-python-sdk/src
```

or add it to the PYTHONPATH at the top of your Python scripts using BaseSpacePy:

```
import sys
sys.path.append('/my/path/BaseSpacePy_vx.x/src')
import BaseSpacePy
```

To test that everything is working as expected, launch a Python prompt and try importing 'BaseSpacePy':

```
mkallberg@ubuntu:~/$ python
>>> import BaseSpacePy
```

2.2 Application triggering

This section demonstrates how to retrieve the `AppLaunch` object produced when a user triggers a BaseSpace App. Further, we cover how to automatically generate the scope strings to request access to the data object (be it a project, a sample, or an analysis) that the App was triggered to analyze.

The initial http request to our App from BaseSpace is identified by an `ApplicationActionId`, using this piece of information we are able to obtain information about the user who launched the App and the data that is sought analyzed by the App. First, we instantiate a BaseSpaceAuth object using the `client_key` and `client_secret` codes provided on the BaseSpace developers website when registering our App:

```
from BaseSpacePy.api.BaseSpaceAuth import BaseSpaceAuth

# initialize an authentication object using the key and secret from your app
# Fill in with your own values
client_key                 = <my key>
client_secret              = <my secret>
ApplicationActionId        = <my action id>
BaseSpaceUrl               = 'https://api.cloud-endor.illumina.com/'
version                    = 'v1pre2/'

# First we will initialize a BaseSpace authentication object
BSauth = BaseSpaceAuth(client_key,client_secret,BaseSpaceUrl,version)

# By supplying the application trigger id we can get out an AppLaunch object
triggerObj = BSauth.getAppTrigger(ApplicationActionId)
print str(triggerObj)
```

```
Output[]:

https://api.cloud-endor.illumina.com/v1pre2/applicationactions/<my action id>
```

We can get the type of object the app was triggered on from the `getLaunchType`-method in the `BaseSpaceAuth` instance:

```
# The trigger type is a list with two items, the first a string taking the one of the values ('Proje
# and the second a list of the objects of that type
triggerType = triggerObj.getLaunchType()
print "\nType of data the app was triggered on"
```

```
print triggerType
print "\nWe can get a handle for the user who triggered the app\n" + str(triggerObj.User)

Output[]:

Type of data the app was triggered on
['Projects', [YourProject]]

We can get a handle for the user who triggered the app
152152: Morten Kallberg
```

To start working, we will want to expand our permission scope for the trigger object so we can read and write data. The details of this process is the subject of the next section. We end this section by demonstrating how one can easily obtain the so-called "scope string," used when requesting further access, from the trigger object:

```
triggerObj = triggerType[1][-1]
print "\nThe scope string for requesting write access to the trigger object is:"
print triggerObj.getAccessStr(scope='write')

Output[]:

The scope string for requesting write access to the trigger object is:
write project 89
```

2.3 Requesting an access-token for data browsing

Here we demonstrate the basic BaseSpace authentication process. The work-flow outlined here is

1. Request of access to a specific data-scope
2. User approval of access request
3. Browsing data

Note: It will be useful if you are logged in to the BaseSpace web-site before launching this example to make the access grant procedure faster.

Again we will start out by initializing a `BaseSpaceAuth` object:

```
from BaseSpacePy.api.BaseSpaceAuth import BaseSpaceAuth
import time

# initialize an authentication object using the key and secret from your app
client_key                = <my key>
client_secret             = <my secret>
BaseSpaceUrl              = 'https://api.cloud-endor.illumina.com/'
version                   = 'v1pre2/'
BSauth = BaseSpaceAuth(client_key,client_secret,BaseSpaceUrl,version)
```

First get verification code and uri for scope 'browse global'

```
deviceInfo = BSauth.getVerificationCode('browse global')
```

At this point the user must visit the verification uri to grant us access

```
## PAUSE HERE
# Have the user visit the verification uri to grant us access
print "Please visit the uri within 30 seconds and grant access"
print deviceInfo['verification_with_code_uri']
time.sleep(30)
## PAUSE HERE
```

```
Output[]:

Please visit the uri within 10 seconds and grant access
https://cloud-endor.illumina.com/oauth/device?code=<my device code>
```

There are two options for obtaining the access-token and instantiating a `BaseSpaceAPI` object:

```
# Get the access-token directly and instantiate an api yourself
#token = BSauth.getAccessToken(deviceInfo['device_code'])
#print "My token " + str(token)

# Alternatively we can generate an access-token and request a BaseSpaceApi instance
# with the newly generated token in one step
myAPI = BSauth.getBaseSpaceApi(deviceInfo['device_code'])
print myAPI
```

```
Output[]:

BaseSpaceAPI instance - using token=<my access token>
```

At this point we can start using the `BaseSpaceAPI` instance to browse the available data for the current user, the details of this process is the subject of the next section. Here we will end with showing how the API object can be used to list all BaseSpace genome instances:

```
# We will get all available genomes with our new api!
allGenomes  = myAPI.getAvailableGenomes()
print "\nGenomes \n" + str(allGenomes)
```

```
Output[]:

Genomes
[Arabidopsis thaliana, Bos Taurus, Escherichia coli, Homo sapiens, Mus musculus, Phix,\
 Rhodobacter sphaeroides, Rattus norvegicus, Saccharomyces cerevisiae, Staphylococcus aureus, Bacillu
```

2.4 Browsing data with `global browse`-access

This section demonstrates basic browsing of BaseSpace objects once an access-token for global browsing has been obtained. We will see how objects can be retrieved using either the `BaseSpaceAPI` class or by use of method calls on related object instances (for example once a `user` instance we can use it to retrieve all project belonging to that user).

First we will initialize a `BaseSpaceAPI` using our access-token for `global browse`:

```
from BaseSpacePy.api.BaseSpaceAPI import BaseSpaceAPI

# REST server information and user access token
server          = 'https://api.cloud-endor.illumina.com/'
version         = 'v1pre2'
access_token    = <my access token>
```

```
# First, create a client for making calls for this user session
myAPI   = BaseSpaceAPI(AccessToken=access_token,apiServer= server + version)
```

First we will try to retrieve a genome object:

```
# Now grab the genome with id=4
myGenome    = myAPI.getGenomeById('4')
print "\nThe Genome is " + str(myGenome)
print "We can get more information from the genome object"
print 'Id: ' + myGenome.Id
print 'Href: ' + myGenome.Href
print 'DisplayName: ' + myGenome.DisplayName

Output[]:

The Genome is Homo sapiens
We can get more information from the genome object
Id: 4
Href: v1pre2/genomes/4
DisplayName: Homo Sapiens - UCSC (hg19)
```

Using a comparable method we can get a list of all available genomes:

```
# Get a list of all genomes
allGenomes  = myAPI.getAvailableGenomes()
print "\nGenomes \n" + str(allGenomes)

Output[]:

Genomes
[Arabidopsis thaliana, Bos Taurus, Escherichia coli, Homo sapiens, Mus musculus, Phix,\
 Rhodobacter sphaeroides, Rattus norvegicus, Saccharomyces cerevisiae, Staphylococcus aureus, Bacillu
```

Now, let us retrieve the `User` objects for the current user, and list all projects for this user:

```
# Take a look at the current user
user        = myAPI.getUserById('current')
print "\nThe current user is \n" + str(user)

# Now list the projects for this user
myProjects   = myAPI.getProjectByUser('current')
print "\nThe projects for this user are \n" + str(myProjects)

Output[]:

The current user is
152152: Morten Kallberg

The projects for this user are
[HiSeq 2500, Bolt, YourProject, 2X151 Rhodobacter Resequencing, EColi resequencing]
```

We can also achieve this by making a call using the `user` instance. Notice that these calls take an instance of `BaseSpaceAPI` with apporpriate priviliges to complete the transaction as parameter, this true for all retrieval method calls made on data objects:

```
myProjects2 = user.getProjects(myAPI)
print "\nProjects retrieved from the user instance \n" + str(myProjects2)

# List the runs available for the current user
```

```
runs = user.getRuns(myAPI)
print "\nThe runs for this user are \n" + str(runs)

Output[]:

Projects retrieved from the user instance
[HiSeq 2500, Bolt, YourProject, 2X151 Rhodobacter Resequencing, EColi resequencing]

The runs for this user are
[2X151 Rhodobacter Resequencing, 2x26 Validation Human 4-Plex, EColi resequencing]
```

In the same manner we can get a list of accessible user runs:

```
runs = user.getRuns(myAPI)
print "\nRuns retrieved from user instance \n" + str(runs)

Output[]:

Runs retrieved from user instance
[2X151 Rhodobacter Resequencing, 2x26 Validation Human 4-Plex, EColi resequencing]
```

2.5 Accessing file-trees and querying BAM/VCF files

In this section we demonstrate how to access samples and analysis from a projects and how to work with the available file data for such instances. In addition, we take a look at some of the special queuring methods associated with BAM- and VCF-files.

Again, start out by initializing a `BaseSpacePy` instance and retrieving all projects belonging to the current user:

```
# First, create a client for making calls for this user session
myAPI    = BaseSpaceAPI(AccessToken=access_token,apiServer= server + version)
user         = myAPI.getUserById('current')
myProjects    = myAPI.getProjectByUser('current')
```

Now we can list all the analyses and samples for these projects

```
for singleProject in myProjects:
    print "# " + str(singleProject)
    analyses = singleProject.getAnalyses(myAPI)
    print "    The analysis for project " + str(singleProject) + " are \n\t" + str(analyses)
    samples = singleProject.getSamples(myAPI)
    print "    The samples for project " + str(singleProject) + " are \n\t" + str(samples)

Output[]:

# HiSeq 2500
    The analysis for project HiSeq 2500 are
        [Resequencing - Completed]
    The samples for project HiSeq 2500 are
        [NA18507]
# Bolt
    The analysis for project Bolt are
        [Amplicon - Completed, Amplicon - Completed, Amplicon ...
    The samples for project Bolt are
        [sample_1, sample_2, sample_3, ...

......
```

We'll take a further look at the files belonging to the sample from the last project in the loop above:

```
for s in samples:
    print "Sample " + str(s)
    ff = s.getFiles(myAPI)
    print ff

Output[]:

Sample Ecoli
[s_G1_L001_R1_001.fastq.1.gz, s_G1_L001_R1_002.fastq.1.gz, s_G1_L001_R2_001.fastq.1.gz, s_G1_L001_R2_
```

Now, have a look at some of the methods calls specific to `Bam` and `VCF` files. First, we will get a `Bam`-file and then retrieve the coverage information available for chromosome 2 between positions 1 and 20000:

```
# Now do some work with files
# we'll grab a BAM by id and get the coverage for an interval + accompanying meta-data
myBam = myAPI.getFileById('2150156')
print myBam
cov     = myBam.getIntervalCoverage(myAPI,'chr2','1','20000')
print cov
covMeta = myBam.getCoverageMeta(myAPI,'chr2')
print covMeta

Output[]:

sorted.bam
Chrchr2: 1-20096: BucketSize=16
CoverageMeta: max=20483 gran=128
```

For `VCF`-files we can filter variant calls based on chromosome and location as well:

```
# and a vcf file
myVCF = myAPI.getFileById('2150158')
# Get the variant meta info
varMeta = myVCF.getVariantMeta(myAPI)
print varMeta
var     = myVCF.filterVariant(myAPI,'2','1', '11000')
print var

Output[]:

VariantHeader: SampleCount=1
[Variant - chr2: 10236 id=['.'], Variant - chr2: 10249 id=['.'], ....
```

2.6 Creating an analysis and uploading results

In this section we will see how to create a new analysis object, change its state and upload result files to it as well as retrieve files from it.

First, create a client for making calls for this user session:

```
myBaseSpaceAPI    = BaseSpaceAPI(AccessToken=access_token,apiServer= server + version)
#
## Now we'll do some work of our own. First get a project to work on
## we'll need write permission, for the project we are working on
## meaning we will need get a new token and instantiate a new BaseSpaceAPI
```

```
p = myBaseSpaceAPI.getProjectById('89')
# A short-cut for getting a scope string if we already have a project-instance:
print p.getAccessStr(scope='write')
# or simply
p.getAccessStr()

Output[]:

write project 89
```

Assuming we now have write access for the project, we will list the current analyses for the project:

```
ana = p.getAnalyses(myBaseSpaceAPI)
print "\nThe current analyses are \n" + str(ana)

Output[]:

The current analyses are
[Results for sample 123 - Working, Results for sample 124 - Working...
```

To create an analysis for a project, simply give the name and description to the `createAnalysis`-method:

```
analysis = p.createAnalysis(myBaseSpaceAPI,"My very first analysis!!","This is my analysis")
print "\nSome info about our new analysis"
print analysis
print analysis.Id
print analysis.Status
# we can change the status of out analysis and add a status-summary as follows
analysis.setStatus(myBaseSpaceAPI,'completed',"We worked hard.")
print "\nAfter a change of status we get\n" + str(analysis)

### List the analyses again and see if our new object shows up
ana = p.getAnalyses(myBaseSpaceAPI)
print "\nThe updated analyses are \n" + str(ana)

Output[]:

Some info about our new analysis
My very first analysis!! - Working
94094
Working

After a change of status we get
My very first analysis!! - Completed

The updated analyses are
[Results for sample 123 - Working, Results for sample 124 - Working, Results for sample 124 - Workin
```

Now we will make another analysis and try to upload some files to it:

```
analysis2 = p.createAnalysis(myBaseSpaceAPI,"My second analysis","This one I will upload to")
analysis2.uploadFile(myBaseSpaceAPI, '/my/file/dir/testFile2.txt', 'BaseSpaceTestFile.txt', '/mydir/'
print "\nMy analysis number 2 \n" + str(analysis2)
#
## Check to see if our new file made it
analysisFiles = analysis2.getFiles(myBaseSpaceAPI)
print "\nThese are the files in the analysis"
print analysisFiles
f = analysisFiles[-1]
```

```
Output[]:

My analysis number 2
My second analysis - Working

These are the files in the analysis
[BaseSpaceTestFile.txt]
```

We can even download our newly uploaded file in the following manner:

```
f.downloadFile(myBaseSpaceAPI,'/path/to/place/file/in/')
```

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *search*

INDEX

I

M