

Package: BaseSpaceR (via r-universe)

October 10, 2026

Title R SDK for BaseSpace RESTful API

Description A rich R interface to Illumina's BaseSpace cloud computing environment, enabling the fast development of data analysis and visualisation tools.

Version 1.56.0

Author Adrian Alexa

Maintainer Jared O'Connell <joconnell@illumina.com>

biocViews Infrastructure, DataRepresentation, ConnectTools, Software, DataImport, HighThroughputSequencing, Sequencing, Genetics

Depends R (>= 2.15.0), RCurl, RJSONIO

Imports methods

Suggests RUnit, IRanges, Rsamtools

License Apache License 2.0

Collate generics.R misc.R ServiceURI.R Error.R AppAuth.R
AppSessionAuth.R Href.R Properties.R Response.R Users.R Runs.R
Genomes.R Projects.R AppResults.R AppSessions.R Samples.R
Files.R Files_extra.R Variants.R Coverage.R zzz.R

Config/pak/sysreqs make

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:37:30 UTC

RemoteUrl <https://github.com/bioc/BaseSpaceR>

RemoteRef RELEASE_3_23

RemoteSha 3904e5a98d9b7033678a25730c2f6c01c0ce3ad4

Contents

BaseSpaceR-package	2
aAuth	3
AppAuth-class	4
AppResults-class	5

AppSessionAuth-class	6
AppSessions-class	7
Coverage	7
Files-class	8
FilesMethods	9
Genomes-class	10
Private methods	12
Projects-class	12
Response-class	13
ResponseStatus-class	15
Runs-class	16
Samples-class	17
ServiceURI-class	18
Users-class	19
Variants	20
Index	22

BaseSpaceR-package	<i>R SDK for BaseSpace RESTful API</i>
--------------------	--

Description

The BaseSpaceR package provides a rich R interface to Illumina's BaseSpace cloud computing environment, enabling the fast development of data analysis and visualisation tools.

Details

BaseSpace is Illumina's next-generation sequencing cloud computing environment designed with biologists in mind. Researcher can easily store, analyze, collaborate, and share genetic data (<https://basespace.illumina.com>).

BaseSpaceR is a SDK offering methods and data structures for working with the data resources BaseSpace REST API exposes.

Features include:

- Persistent connection with the REST server.
- Support for the REST API query parameters.
- Vectorized operations in line with the R semantic. Allows for queries across multiple Projects, Samples, AppResults, Files, etc.
- S4 class system used to represent the BaseSpace data model.
- Portability on most platforms: Linux, Windows and Mac OS X.

Author(s)

Adrian Alexa

Maintainer: Adrian Alexa <aalexa@illumina.com>

References

BaseSpace <https://basespace.illumina.com>

BaseSpace API <https://developer.basespace.illumina.com>

See Also

[AppAuth-class](#) for details on how to establish a connection with the BaseSpace server.

[Response-class](#) for details on the structure of the objects and the interface used by the API.

[Users-class](#), [Genomes-class](#), [Runs-class](#), [Projects-class](#), [Samples-class](#), [Files-class](#) and [AppResults-class](#) for details on various resources supported by the API.

aAuth

Sample AppAuth instance with 'browse global' scope

Description

The aAuth is an instance of a AppAuth object. It can be used to browse some of the public resources available in BaseSpace without requiring user authentication.

Usage

```
data(aAuth)
```

Source

Build from a pre-generated access token. See [AppAuth-class](#) for code examples on how-to generate such an object.

Examples

```
data(aAuth)

## print the object
aAuth
```

AppAuth-class

AppAuth *class***Description**

Class to manage client's communication and permissions with BaseSpace REST serve.

Methods

initializeAuth signature(x = "AppAuth"): ...

requestAccessToken signature(x = "AppAuth"): ...

hasAccess signature(x = "AppAuth"): ...

Author(s)

Adrian Alexa

References

<https://developer.basespace.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[Runs](#), [Projects](#), [Samples](#), etc. for examples of objects using the AppAuth handler.

[ServiceURI](#) for a low-level object managing the REST calls.

Examples

```
showClass("AppAuth")

## load an AppAuth instance containing a pre-generated access token
data(aAuth)
aAuth

hasAccess(aAuth)

## new AppAuth instance using a pre-generated access token
my_access_token <- "eee44c28ba0e43a1badb85c5ce7bb94d"
myHandle <- AppAuth(access_token = my_access_token)
myHandle

Users(myHandle)

## using the OAuth v2 workflow
## Not run:
## paste your client_id and client_secret here
myAppClientId = ""
```

```
myAppClientSecret = ""

## instantiate a new AppAuth object
myHandle <- AppAuth(client_id = myAppClientId,
                    client_secret = myAppClientSecret,
                    scope = "browse global")

## Open the showed URI in a browser and perform the authentication.

requestAccessToken(myHandle)
hasAccess(myHandle)

## End(Not run)
```

AppResults-class

AppResults and "AppResultsSummary" objects

Description

Classes and methods to handle the AppResults resource.

Author(s)

Adrian Alexa

References

<https://developer.basespace.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[AppAuth](#) and [Projects](#).

Examples

```
showClass("AppAuth")
```

AppSessionAuth-class AppSessionAuth *class*

Description

Extension of the AppAuth class...

Author(s)

Adrian Alexa

References

<https://developer.basespace.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[Runs](#), [Projects](#), [Samples](#), etc. for examples of objects using the AppAuth handler.

[ServiceURI](#) for a low-level object managing the REST calls.

Examples

```
showClass("AppAuth")

## using the OAuth v2 workflow
## Not run:
## paste your client_id and client_secret here
myAppClientId = ""
myAppClientSecret = ""

## instantiate a new AppAuth object
myHandle <- AppAuth(client_id = myAppClientId,
                    client_secret = myAppClientSecret,
                    scope = "browse global")

## Open the showed URI in a browser and perform the authentication.

requestAccessToken(myHandle)
hasAccess(myHandle)

## End(Not run)
```

AppSessions-class	AppSessions <i>object</i>
-------------------	---------------------------

Description

Classes and methods to handle the AppSessions resource.

Details

coming soon...

Author(s)

Adrian Alexa

References

<https://developer.basespace.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[AppAuth](#) and [Response](#).

Examples

```
showClass("AppAuth")
```

Coverage	<i>Methods for accessing coverage data from BAM files</i>
----------	---

Description

This methods are used to provide mean read coverage depth in a particular chromosomal region.

Usage

```
getCoverageStats(x, ...)  
getCoverage(x, ...)
```

Arguments

- x** An object of class AppAuth.
- ...** Additional arguments supported by the REST API.
 - id** File id (for a BAM file) within BaseSpace.
 - chrom** Character string given the chromosome name (UCSC).

Details

Coming soon...

Value

These methods return a list with a representation of coverage histogram.

Author(s)

Adrian Alexa

References

<https://developer.basespace.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[AppAuth](#).

Examples

```
data(aAuth)

## get the Ids of some BAM files
##...

## You might require Read access to the AppResult
#initializeAuth(aAuth, scope = paste("browse global, read project", 12))
#requestAccessToken(aAuth)

#getCovStats(aAuth, id = bid, "phix")

#readcov <- getCov(aAuth, id = bid, "phix", StartPos = 1L, EndPos = 5e3L)[[1]]
#barplot(readcov$MeanCoverage, col = "lightblue1", border = NA)
#plot(readcov$MeanCoverage, col = "lightblue2", type = "l", lwd = 2)
```

Files-class

Files and FilesSummary objects

Description

Classes and methods to handle the Files resource.

Details

The Files resource provides access to files stored in BaseSpace. A file should be seen as a data stream and associated attributes (date created, size, type, etc.).

Files are associated with specific Runs, Samples, or AppResults and the Files resource provides the interface for manipulating these files.

Author(s)

Adrian Alexa

References

<https://developer.basespace.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[AppAuth](#), [Runs](#), [Samples](#) and [AppResults](#).

Examples

```
data(aAuth)

## get one AppResult
reseq <- AppResults(listAppResults(aAuth, projectId = 21383369, Limit = 1))

f <- listFiles(reseq)
TotalCount(f)
Name(f)

identical(f, listFiles(aAuth, appResultId = Id(reseq)))

## list only the BAM files
f <- listFiles(aAuth, appResultId = Id(reseq), Extensions = ".bam")
Name(f)
```

Description

Methods to handle various files types.

Details

The Files resource provides access to files stored in BaseSpace. A file should be seen as a data stream and associated attributes (date created, size, type, etc.).

For known file types (.bam, .vcf, etc.) we offer wrappers to 'map' this objects to various Bioconductor objects.

Author(s)

Adrian Alexa

References

<https://developer.basespace.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[AppAuth](#), [AppResults](#) and [Files](#)

Examples

```
data(aAuth)

## get one AppResult
reseq <- AppResults(listAppResults(aAuth, projectId = 21383369, Limit = 1))

f <- listFiles(reseq)
TotalCount(f)
Name(f)

## list only the BAM files
f <- listFiles(aAuth, appResultId = Id(reseq), Extensions = ".bam")
Name(f)

## construct the BAMFile
##...
```

Genomes-class

Genomes *and* GenomesSummary *objects*

Description

Classes and methods to handle the Genomes resource.

Browsing

`listGenomes(x, ...)` lists all the available genomes, returning only a small summary for each genome.

`x` is an `AppAuth` object.

`...` Parameters supported by the REST API, specified as `name = value`. For example, `listGenomes(aAuth, Offset = 5, Limit = 2)`

Constructor

`Genomes()`: Instantiates an empty `Genomes` object. Same as `new("Genomes")`.

`Genomes(x, id)`: `x` is an `AppAuth` object. `id` is either an integer or a character string storing an integer. `id` can have length larger than 1.

`Genomes(x)`: `x` is an `GenomesSummary` object.

Author(s)

Adrian Alexa

References

<https://developer.basemap.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[AppAuth](#), [Response](#).

Examples

```
data(aAuth)

## list all available genomes
g <- listGenomes(aAuth)
g$SpeciesName

## using the REST API query parameters
listGenomes(aAuth, Limit = 2)
g <- listGenomes(aAuth, Offset = 5, Limit = 2, SortBy = "Build")
g

## get the details for the listed genomes
Genomes(g)

## get the genomes based on their ID
Genomes(aAuth, id = 4)
## if the ID is missing then NULL is returned for that particular ID
Genomes(aAuth, id = c(4, 1, 110))
```

Private methods

Internal methods

Description

This page is used as a placeholder for private methods in order to pass the R checks.

Details

Internal use.

Author(s)

Adrian Alexa

Projects-class

Projects and ProjectsSummary objects

Description

Classes and methods to handle the Projects resource.

Details

The Projects resource provides a logical grouping of the Samples resource and the AppResults resource for a given user.

Browsing

`listProjects(x, ...)` lists all the available projects visible to the user, returning only a small summary for each project.

`x` is an AppAuth object.

`...` Parameters supported by the REST API, specified as `name = value`. For example, `listProjects(x, Limit = 2)`

Constructor

`Projects()`: Instantiates an empty Projects object. Same as `new("Projects")`.

`Projects(x, id)`: `x` is an AppAuth object. `id` is either an integer or a character string storing an integer. `id` can have length larger than 1.

`Projectcs(x)`: `x` is an ProjectsSummary object.

Author(s)

Adrian Alexa

References

<https://developer.basespace.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[AppAuth](#), [Response](#), [Users](#) and [Runs](#).

Examples

```
data(aAuth)

p <- listProjects(aAuth)
p

Projects(aAuth, id = c(2, 12, 1012))
Projects(p)

## Make a new project ...
createProject(aAuth, name = "My Project X")

## We need 'write global' access to be able to create a new project
## Not run:
initializeAuth(aAuth, scope = "write global")
requestAccessToken(aAuth)

createProject(aAuth, name = "My Project X")

## End(Not run)
```

Response-class	Item, Collection <i>and</i> Response <i>objects</i>
----------------	---

Description

These virtual classes provide the building blocks for the containers return by quering various BaseSpace resource. They are modeled after the Response container defined by the REST API.

Conceptually there are two response types exposed by BaseSpace, an individual response and a collection response. The former, modeled by the Item S4 class, is used when querying an individual item/instance within a resource. The later is used for listing the items/instances available for a given resource and is modeled by the Collection S4 class.

Item Accessors

The `Item` class models a simple unordered set of key/value pairs. There is a core set of keys, for which access methods are defined and which are inherited by any child class.

In the following `x` is an `Item` object.

\$: The `$` operator can be used to access the `Item` elements. When `x` is an `Item` object this is equivalent to `@`. But one can think at it as user level operator. Unlike `@` the `replace` method is not implemented for `$`.

Id(x): Id of the resource. Character string, though it will always be an integer.

Name(x): Name of the selected resource.

Href(x): Location of the resource in the API. The first component of the URI is the version of the REST API.

DateCreated(x): When this resource was created. Character string. It can be converted to a `Date` instance by `as.Date(DateCreated(x))`.

UserOwnedBy(x): Information about the User who owns this resource. At this moment this is a `list` object, but it might be replaced with an object at a future point.

Status(x): The status of the resource. Can be of any type and it will be defined by the classes extended `Item`.

HrefBaseSpaceUI(x): The location of this project in `BaseSpace`. Character string giving the complete URL within the `BaseSpace` dashboard.

Collection Accessors

The `Collection` class models an ordered set of `Item` objects and a set of predefined attributes. The interface provided by the `Item` class is implemented by this class. However, since we deal with an ordered set of objects, the methods and the access methods, return a vector of the same length as the size of the collection.

In the following `x` is a `Collection` object.

All accessor implemented by the `item` class are implemented by `Collection` class. However here the return value is a vector. If `x` has 2 elements, then `Id(x)` will be a vector with 2 elements. The same stands for the general accessor `$`.

length(x): Returns the number of elements in `Collection x`.

Items(x): List of `Item` objects.

TotalCount(x): The total number of items in the collection as reported by the queried resource.

Offset(x): The starting point the collection was read from.

Limit(x): The maximum number of items returned. Ranges from 0 to 1024.

SortDir(x): The way the collection is sorted, either ascending or descending.

SortBy(x): The field to use to sort the collection.

Subsetting

Comming soon ...

Methods

`length(x)`: Returns the number of elements in Collection x.

`show(x)`: Prints the object.

`as.list(x)`: R list representation of the object.

Author(s)

Adrian Alexa

Examples

```
showClass("Item")
showClass("Collection")
showClass("Response")
```

ResponseStatus-class *Class* "ResponseStatus"

Description

The ResponseStatus class is used internally to manage the status messages returned by the REST service.

Objects from the Class

Objects can be created by calls of the form `ResponseStatus()`. `ResponseStatus()` creates an `ResponseStatus` instances...

Methods

success `signature(x = "ServiceURI")`: Returns TRUE if the HTTP status is 2xx.

show `signature(object = "ServiceURI")`: ...

Author(s)

Adrian Alexa

Examples

```
showClass("ResponseStatus")
```

Runs-class

Runs *and* RunSummary *objects*

Description

Classes and methods to handle the Runs resource.

Details

The Runs resource contains the raw data produced by the instruments, the base calls, together with run metrics, instrument health data, and other information used for data processing and analysis.

Browsing

`listRuns(x, ...)` lists all the available runs visible to the user, returning only a small summary for each run.

`x` is an AppAuth object.

`...` Parameters supported by the REST API, specified as `name = value`. For example, `listRuns(x, Limit = 2, Statuses = "Failed")`

Constructor

`Runs()`: Instantiates an empty Runs object. Same as `new("Runs")`.

`Runs(x, id)`: `x` is an AppAuth object. `id` is either an integer or a character string storing an integer. `id` can have length larger than 1.

`Runs(x)`: `x` is a RunSummary object.

Author(s)

Adrian Alexa

References

<https://developer.baspace.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[AppAuth](#), [Response](#), and [Users](#).

Examples

```
data(aAuth)

r <- listRuns(aAuth)
r

listRuns(aAuth, Statuses = "Failed") # no failed runs
```

```
listRuns(aAuth, Statuses = "Complete")
listRuns(aAuth, SortBy = "Id", SortDir="Desc")

Runs(r)[[1]]

Runs(aAuth, id = 101102)
Runs(r)

Runs(aAuth, id = c(Id(r), "11111")) # the third element must be NULL
```

Samples-class

Samples and SamplesSummary objects

Description

Classes and methods to handle the Samples resource.

Details

In general samples are the result of demultiplexing and are thought as the holding the input data for an App. One example of data within a Samples resource are the FASTQ files.

Browsing

`listSamples(x, projectId, ...)` lists all the available samples associated with a particular project. It returns a small summary for each existing sample.

x is an AppAuth object.

projectId the ID of the project we want to explore.

... Parameters supported by the REST API, specified as name = value. For example, `listSamples(x, projectId = "1", Limit = 2)`

`listSamples(x, ...)`

x is an Projects object.

... Parameters supported by the REST API.

Constructor

`Samples()`: Instantiates an empty Samples object. Same as `new("Samples")`.

`Samples(x, id)`: **x** is an AppAuth object. **id** is either an integer or a character string storing an integer. **id** can have length larger than 1.

`Samples(x)`: **x** is an SamplesSummary object.

Author(s)

Adrian Alexa

References

<https://developer.basespace.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[AppAuth](#) and [Projects](#).

Examples

```
data(aAuth)

## list all the available projects and select one
p <- Projects(listProjects(aAuth, Limit = 1), simplify = TRUE)
p

## list the samples available in this project
allS <- listSamples(aAuth, projectId = Id(p))

## we can call listSamples() directly using 'p'
identical(allS, listSamples(p))

oneS <- listSamples(aAuth, projectId = Id(p), Limit = 1)
oneS
Samples(oneS) # list with one Samples object
Samples(oneS, simplify = TRUE) # Samples object
```

ServiceURI-class

Class "ServiceURI"

Description

The ServiceURI class is a general container for storing the URI of a REST based Web service.

Objects from the Class

Objects can be created by calls of the form `ServiceURI()`. `ServiceURI()` creates an `ServiceURI` instances out of an URL and an API resource.

Slots

`url`: Object of class "character" ~~
`version`: Object of class "character" ~~

Methods

show signature(object = "ServiceURI"): ...
uri signature(x = "ServiceURI"): ...

Author(s)

Adrian Alexa

Examples

```
showClass("ServiceURI")
```

Users-class

Users *objects and resources*

Description

Class to manage the Users resource.

Details

The Users resource allows the client to get basic information about the user that is currently using the application.

To query this resource we use the Users() method.

Constructor

Users(): Instantiates an empty Users object. Same as new("Users").

Users(x): x is an AppAuth object. The function returns a Response object of class Users.

Author(s)

Adrian Alexa

References

<https://developer.basespace.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[AppAuth](#), [Response](#).

Examples

```
data(aAuth)

## Empty instance
Users()

## Querying the Users resource using the AppAuth handler
u <- Users(aAuth)
## Printing the object shows the response elements
u
```

```

## Accesors
Id(u)
Name(u)
## Using the general '$' accesor, same interface as 'list'
u$Id
u$Email # there is no accesor method Email(), so '$' is useful here!
u$fakeElement # returns NULL (to keep the same semantic as 'list')

## Quering the resource unsing a Response object
Users(u) # u is of class Users which extends Response

## Specifying a user ID. ID can be specify either as an integer or as a string
Users(aAuth, id = 1463464) # must work if given as an integer(even of mode numeric)
Users(aAuth, id = "1463464") # must work

## This should fail since is not the current user
tryCatch(Users(aAuth, id = "660666"), error = function(e) cat("No access to this user data!\n"))

```

Variants

Methods for accessing varinant data

Description

Manipulating variant data.

Usage

```

getVariantSet(x, ...)
getVariants(x, ...)

```

Arguments

x An object of class AppAuth.

... Additional arguments supported by the REST API.

id File id (for a BAM file) within BaseSpace.

chrom Character string given the chromosome name (UCSC).

Details

Coming soon...

Value

These methods return a list with a representation of variant data.

Author(s)

Adrian Alexa

References

<https://developer.basespace.illumina.com/docs/content/documentation/rest-api/api-reference>

See Also

[AppAuth](#).

Examples

```
data(aAuth)

## get the ids of VCF files (within an AppResult? )
reseq <- listAppResults(aAuth, projectId = 21383369, Limit = 1)
AppResults(reseq)

vcfs <- listFiles(AppResults(reseq), Extensions = ".vcf")
Name(vcfs)
Id(vcfs)
vcfs

## Not run:
## variant metadata
getVariantSet(aAuth, vid)

## get the variants
getVariants(aAuth, vid, chrom = "chr")

v <- getVariants(aAuth, vid, chrom = "chr", EndPos = 1000000L, Limit = 5)
v

## End(Not run)
```

Index

* classes

AppAuth-class, [4](#)
 AppResults-class, [5](#)
 AppSessionAuth-class, [6](#)
 AppSessions-class, [7](#)
 Files-class, [8](#)
 Genomes-class, [10](#)
 Private methods, [12](#)
 Projects-class, [12](#)
 Response-class, [13](#)
 ResponseStatus-class, [15](#)
 Runs-class, [16](#)
 Samples-class, [17](#)
 ServiceURI-class, [18](#)
 Users-class, [19](#)

* datasets

aAuth, [3](#)

* methods

AppAuth-class, [4](#)
 AppResults-class, [5](#)
 AppSessionAuth-class, [6](#)
 AppSessions-class, [7](#)
 Coverage, [7](#)
 Files-class, [8](#)
 FilesMethods, [9](#)
 Genomes-class, [10](#)
 Private methods, [12](#)
 Projects-class, [12](#)
 Response-class, [13](#)
 ResponseStatus-class, [15](#)
 Runs-class, [16](#)
 Samples-class, [17](#)
 ServiceURI-class, [18](#)
 Users-class, [19](#)

* misc

Variants, [20](#)

* package

BaseSpaceR-package, [2](#)

[, Collection-method (Response-class), [13](#)

[, Response-method (Response-class), [13](#)

[[, Collection, numeric-method
 (Response-class), [13](#)

[[, Response, ANY-method
 (Response-class), [13](#)

\$.Collection-method (Response-class), [13](#)

\$.Item-method (Response-class), [13](#)

\$.Response-method (Response-class), [13](#)

aAuth, [3](#)

AppAuth, [5](#), [7–11](#), [13](#), [16](#), [18](#), [19](#), [21](#)

AppAuth (AppAuth-class), [4](#)

AppAuth-class, [3](#), [4](#)

appResultCollection (AppResults-class),
[5](#)

appResultCollection-class
 (AppResults-class), [5](#)

appResultItem (AppResults-class), [5](#)

appResultItem-class (AppResults-class),
[5](#)

AppResults, [9](#), [10](#)

AppResults (AppResults-class), [5](#)

AppResults, AppAuth-method
 (AppResults-class), [5](#)

AppResults, AppResultsSummary-method
 (AppResults-class), [5](#)

AppResults, missing-method
 (AppResults-class), [5](#)

AppResults-class, [3](#), [5](#)

AppResultsSummary-class
 (AppResults-class), [5](#)

AppSessionAuth-class, [6](#)

appSessionItem (AppSessions-class), [7](#)

appSessionItem-class
 (AppSessions-class), [7](#)

AppSessions (AppSessions-class), [7](#)

AppSessions, AppAuth-method
 (AppSessions-class), [7](#)

AppSessions, AppResults-method
 (AppSessions-class), [7](#)

- AppSessions-class, [7](#)
- as.character, staticHref-method
(Private methods), [12](#)
- as.list, Collection-method
(Response-class), [13](#)
- as.list, Item-method (Response-class), [13](#)
- as.list, Response-method
(Response-class), [13](#)
- auth (Response-class), [13](#)
- auth, Href-method (Private methods), [12](#)
- auth, Response-method (Response-class),
[13](#)
- authNativeClient
(AppSessionAuth-class), [6](#)
- authWebClient (AppSessionAuth-class), [6](#)

- BaseSpaceR (BaseSpaceR-package), [2](#)
- BaseSpaceR-package, [2](#)

- Collection-class (Response-class), [13](#)
- countAppResults (AppResults-class), [5](#)
- countAppResults, AppAuth-method
(AppResults-class), [5](#)
- countAppResults, Projects-method
(AppResults-class), [5](#)
- countAppResults, ProjectsSummary-method
(AppResults-class), [5](#)
- countAppResults, Response-method
(AppResults-class), [5](#)
- countAppSessions (AppSessions-class), [7](#)
- countAppSessions, AppAuth-method
(AppSessions-class), [7](#)
- countAppSessions, Response-method
(AppSessions-class), [7](#)
- countFiles (Files-class), [8](#)
- countFiles, AppAuth-method
(Files-class), [8](#)
- countFiles, AppResults-method
(Files-class), [8](#)
- countFiles, AppResultsSummary-method
(Files-class), [8](#)
- countFiles, Response-method
(Files-class), [8](#)
- countFiles, Runs-method (Files-class), [8](#)
- countFiles, RunsSummary-method
(Files-class), [8](#)
- countFiles, Samples-method
(Files-class), [8](#)
- countFiles, SamplesSummary-method
(Files-class), [8](#)
- countGenomes (Genomes-class), [10](#)
- countProjects (Projects-class), [12](#)
- countProjects, AppAuth-method
(Projects-class), [12](#)
- countProjects, Response-method
(Projects-class), [12](#)
- countRuns (Runs-class), [16](#)
- countRuns, AppAuth-method (Runs-class),
[16](#)
- countRuns, Response-method (Runs-class),
[16](#)
- countSamples (Samples-class), [17](#)
- countSamples, AppAuth-method
(Samples-class), [17](#)
- countSamples, Projects-method
(Samples-class), [17](#)
- countSamples, ProjectsSummary-method
(Samples-class), [17](#)
- countSamples, Response-method
(Samples-class), [17](#)
- Coverage, [7](#)
- createAppResults (AppResults-class), [5](#)
- createAppResults, AppAuth-method
(AppResults-class), [5](#)
- createAppResults, Projects-method
(AppResults-class), [5](#)
- createProject (Projects-class), [12](#)
- createProject, AppAuth-method
(Projects-class), [12](#)

- DateCreated (Response-class), [13](#)
- DateCreated, Collection-method
(Response-class), [13](#)
- DateCreated, Item-method
(Response-class), [13](#)
- DateCreated, Response-method
(Response-class), [13](#)
- DisplayedCount (Response-class), [13](#)
- DisplayedCount, Collection-method
(Response-class), [13](#)
- DisplayedCount, Item-method
(Response-class), [13](#)
- DisplayedCount, Response-method
(Response-class), [13](#)

- fileCollection (Files-class), [8](#)
- fileCollection-class (Files-class), [8](#)

- fileItem (Files-class), 8
- fileItem-class (Files-class), 8
- Files, 10
- Files (Files-class), 8
- Files, AppAuth-method (Files-class), 8
- Files, FilesSummary-method (Files-class), 8
- Files, missing-method (Files-class), 8
- Files-class, 3, 8
- FilesMethods, 9
- FilesSummary-class (Files-class), 8

- genomeCollection (Genomes-class), 10
- genomeCollection-class (Genomes-class), 10
- genomeItem (Genomes-class), 10
- genomeItem-class (Genomes-class), 10
- Genomes (Genomes-class), 10
- Genomes, AppAuth-method (Genomes-class), 10
- Genomes, GenomesSummary-method (Genomes-class), 10
- Genomes, missing-method (Genomes-class), 10
- Genomes-class, 3, 10
- GenomesSummary-class (Genomes-class), 10
- getBAMs (FilesMethods), 9
- getBAMs, AppResults-method (FilesMethods), 9
- getCoverage (Coverage), 7
- getCoverage, AppAuth-method (Coverage), 7
- getCoverageStats (Coverage), 7
- getCoverageStats, AppAuth-method (Coverage), 7
- getFiles (Files-class), 8
- getFiles, AppAuth-method (Files-class), 8
- getIndexedBam (FilesMethods), 9
- getVariants (Variants), 20
- getVariants, AppAuth-method (Variants), 20
- getVariantSet (Variants), 20
- getVariantSet, AppAuth-method (Variants), 20

- hasAccess (AppAuth-class), 4
- hasAccess, AppAuth-method (AppAuth-class), 4
- Href (Response-class), 13
- Href, Collection-method (Response-class), 13
- Href, Item-method (Response-class), 13
- Href, Response-method (Response-class), 13
- HrefBaseSpaceUI (Response-class), 13
- HrefBaseSpaceUI, Collection-method (Response-class), 13
- HrefBaseSpaceUI, Item-method (Response-class), 13
- HrefBaseSpaceUI, Response-method (Response-class), 13

- Id (Response-class), 13
- Id, Collection-method (Response-class), 13
- Id, Item-method (Response-class), 13
- Id, Response-method (Response-class), 13
- initializeAuth (AppAuth-class), 4
- initializeAuth, AppAuth-method (AppAuth-class), 4
- Item-class (Response-class), 13
- Items (Response-class), 13
- Items, Collection-method (Response-class), 13
- Items, Item-method (Response-class), 13
- Items, Response-method (Response-class), 13

- length, Collection-method (Response-class), 13
- length, Item-method (Response-class), 13
- length, Response-method (Response-class), 13
- Limit (Response-class), 13
- Limit, Collection-method (Response-class), 13
- Limit, Item-method (Response-class), 13
- Limit, Response-method (Response-class), 13

- listAppResults (AppResults-class), 5
- listAppResults, AppAuth-method (AppResults-class), 5
- listAppResults, Projects-method (AppResults-class), 5
- listAppSessions (AppSessions-class), 7
- listAppSessions, AppAuth-method (AppSessions-class), 7
- listFiles (Files-class), 8

- listFiles, AppAuth-method (Files-class), 8
- listFiles, AppResults-method (Files-class), 8
- listFiles, AppResultsSummary-method (Files-class), 8
- listFiles, Runs-method (Files-class), 8
- listFiles, RunsSummary-method (Files-class), 8
- listFiles, Samples-method (Files-class), 8
- listFiles, SamplesSummary-method (Files-class), 8
- listGenomes (Genomes-class), 10
- listGenomes, AppAuth-method (Genomes-class), 10
- listGenomes, Response-method (Genomes-class), 10
- listProjects (Projects-class), 12
- listProjects, AppAuth-method (Projects-class), 12
- listProjects, Response-method (Projects-class), 12
- listRuns (Runs-class), 16
- listRuns, AppAuth-method (Runs-class), 16
- listRuns, Response-method (Runs-class), 16
- listSamples (Samples-class), 17
- listSamples, AppAuth-method (Samples-class), 17
- listSamples, Projects-method (Samples-class), 17

- Name (Response-class), 13
- Name, Collection-method (Response-class), 13
- Name, Item-method (Response-class), 13
- Name, Response-method (Response-class), 13

- Offset (Response-class), 13
- Offset, Collection-method (Response-class), 13
- Offset, Item-method (Response-class), 13
- Offset, Response-method (Response-class), 13

- performOAuth (AppAuth-class), 4
- Private methods, 12

- projectCollection (Projects-class), 12
- projectCollection-class (Projects-class), 12
- projectItem (Projects-class), 12
- projectItem-class (Projects-class), 12
- Projects, 4–6, 18
- Projects (Projects-class), 12
- Projects, AppAuth-method (Projects-class), 12
- Projects, missing-method (Projects-class), 12
- Projects, ProjectsSummary-method (Projects-class), 12
- Projects-class, 3, 12
- ProjectsSummary-class (Projects-class), 12
- putFiles (Files-class), 8
- putFiles, AppAuth-method (Files-class), 8

- requestAccessToken (AppAuth-class), 4
- requestAccessToken, AppAuth-method (AppAuth-class), 4
- Response, 7, 11, 13, 16, 19
- Response-class, 3, 13
- ResponseStatus (ResponseStatus-class), 15
- ResponseStatus-class, 15
- runCollection (Runs-class), 16
- runCollection-class (Runs-class), 16
- runItem (Runs-class), 16
- runItem-class (Runs-class), 16
- Runs, 4, 6, 9, 13
- Runs (Runs-class), 16
- Runs, AppAuth-method (Runs-class), 16
- Runs, missing-method (Runs-class), 16
- Runs, RunsSummary-method (Runs-class), 16
- Runs-class, 3, 16
- RunsSummary-class (Runs-class), 16

- sampleCollection (Samples-class), 17
- sampleCollection-class (Samples-class), 17
- sampleItem (Samples-class), 17
- sampleItem-class (Samples-class), 17
- Samples, 4, 6, 9
- Samples (Samples-class), 17
- Samples, AppAuth-method (Samples-class), 17

- Samples,missing-method (Samples-class),
17
- Samples,SamplesSummary-method
(Samples-class), 17
- Samples-class, 3, 17
- SamplesSummary-class (Samples-class), 17
- ServiceURI, 4, 6
- ServiceURI (ServiceURI-class), 18
- ServiceURI-class, 18
- show,Collection-method
(Response-class), 13
- show,Href-method (Private methods), 12
- show,Item-method (Response-class), 13
- show,Response-method (Response-class),
13
- show,ResponseStatus-method
(ResponseStatus-class), 15
- show,ServiceURI-method
(ServiceURI-class), 18
- show,staticHref-method (Private
methods), 12
- SortBy (Response-class), 13
- SortBy,Collection-method
(Response-class), 13
- SortBy,Item-method (Response-class), 13
- SortBy,Response-method
(Response-class), 13
- SortDir (Response-class), 13
- SortDir,Collection-method
(Response-class), 13
- SortDir,Item-method (Response-class), 13
- SortDir,Response-method
(Response-class), 13
- Status (Response-class), 13
- Status,Collection-method
(Response-class), 13
- Status,Item-method (Response-class), 13
- Status,Response-method
(Response-class), 13

- TotalCount (Response-class), 13
- TotalCount,Collection-method
(Response-class), 13
- TotalCount,Item-method
(Response-class), 13
- TotalCount,Response-method
(Response-class), 13

- updateAppSessions (AppSessions-class), 7
- updateAppSessions,AppAuth-method
(AppSessions-class), 7
- uri (ServiceURI-class), 18
- uri,ServiceURI-method
(ServiceURI-class), 18
- userItem (Users-class), 19
- userItem-class (Users-class), 19
- UserOwnedBy (Response-class), 13
- UserOwnedBy,Collection-method
(Response-class), 13
- UserOwnedBy,Item-method
(Response-class), 13
- UserOwnedBy,Response-method
(Response-class), 13
- Users, 13, 16
- Users (Users-class), 19
- Users,AppAuth-method (Users-class), 19
- Users,missing-method (Users-class), 19
- Users,Response-method (Users-class), 19
- Users-class, 3, 19

- Variants, 20