Package: tiledbcloud (via r-universe)

October 7, 2024

Type Package

Title TileDB Cloud Platform R Client Package

Version 0.0.11

Description The TileDB Cloud Platform API Client Package offers access to the TileDB Cloud service.

URL https://github.com/TileDB-Inc/TileDB-Cloud-R

BugReports https://github.com/TileDB-Inc/TileDB-Cloud-R/issues

Depends R (>= 3.3)

License MIT + file LICENSE

Suggests testthat, tinytest, rmarkdown, knitr, arrow, tiledb

Imports jsonlite, httr, R6, base64enc, future

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.3

Repository https://tiledb-inc.r-universe.dev

RemoteUrl https://github.com/TileDB-Inc/TileDB-Cloud-R

RemoteRef HEAD

RemoteSha b0cea6660c6aa68e3cf80abb64c3563a23acf4ca

Contents

`.get_decoded_response_body_or_stop` *Package-internal HTTP-response helper*

Description

This is a package-internal function for code-deduplication within various manual-layer functions.

Usage

```
.get_decoded_response_body_or_stop(
  resultObject,
  result_format,
  entire_json_is_result = FALSE
)
```

Arguments

`resultObject`	Should be a return value from an API function which uses `.wrap_as_api_response` internally. These are functions which are manually edited after OpenAPI auto-gen.
`entire_json_is_result`	If false, return the `"value"` field from the JSON object. This is the right thing to do for returns from the REST server for almost all cases. The true case is only for getting the results from invoking registered Python UDFs from R, in which case the JSON result in its entirety is the UDF output.

Details

It wraps `.get_raw_response_body_or_stop` by decoding the raw response body using any of the three result-format types we support for UDFs. It's a keystroke-saving wrapper around `.get_raw_response_body_or_stop`.

Value

The argument, decoded according to the specified result format.

`.get_empty_response_body_or_stop`

Package-internal HTTP-response helper

Description

This is a package-internal function for code-deduplication within various manual-layer functions.

Usage

```
.get_empty_response_body_or_stop(resultObject)
```

Arguments

`resultObject`	Should be a return value from an API function which uses `.wrap_as_api_response` internally. These are functions which are manually edited after OpenAPI auto-gen.

Details

This wraps `.get_raw_response_body_or_stop`, doing `stop` if there is an API-response error, or if the response is not the empty string. This is for API functions where the expected response is the empty string.

Value

Invisible on success, or `stop()` on failure.

.get_raw_response_body_or_stop *Package-internal HTTP-response helper*

Description

This is a package-internal function for code-deduplication within various manual-layer functions.

Usage

```
.get_raw_response_body_or_stop(resultObject)
```

Arguments

`resultObject`	Should be a return value from an API function which uses `.wrap_as_api_response` internally. These are functions which are manually edited after OpenAPI autogen.

Details

For the API-level functions which use `.wrap_as_api_response`, manual-layer functions will receive either (a) the raw HTTP body, if the `status_code` was 2xx, or (b) an `ApiResponse` object. Using this function, callsites can get the HTTP body (if available), else an informative `stop()`.

Value

The argument, as long as it's of type `raw`. Else, stops. The caller can then decode the raw body.

.wrap_as_api_response *Package-internal HTTP-response helper*

Description

Used for overriding the OpenAPI-autogenerated HTTP-response handling.

Usage

```
.wrap_as_api_response(resp)
```

Arguments

`resp`	A `response` S3 object e.g. from `httr:GET`.

Details

Makes the handling compatible with TileDB REST-server response format; surfaces error messages back to the user. Not intended to be exported from this package; for package-internal use only.

Value

An object of type ApiResponse. In the success case (HTTP 2xx) its content slot is the raw HTTP body. In the failure case (otherwise) its content slot is error-text from the server.

ActivityEventType *ActivityEventType*

Description

ActivityEventType Class

Format

An R6Class generator object

Methods

Public methods:

- ActivityEventType$new()
- ActivityEventType$toJSON()
- ActivityEventType$fromJSON()
- ActivityEventType$toJSONString()
- ActivityEventType$fromJSONString()
- ActivityEventType$clone()

Method new():

Usage:

```
ActivityEventType$new(...)
```

Method toJSON():

Usage:

```
ActivityEventType$toJSON()
```

Method fromJSON():

Usage:

```
ActivityEventType$fromJSON(ActivityEventTypeJson)
```

Method toJSONString():

Usage:

```
ActivityEventType$toJSONString()
```

Method fromJSONString():

Usage:

```
ActivityEventType$fromJSONString(ActivityEventTypeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ActivityEventType$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ApiClient` *ApiClient*

Description

ApiClient Class

Format

An `R6Class` generator object

Details

ApiClient Class

Generic API client for OpenAPI client library builds. OpenAPI generic API client. This client handles the client- server communication, and is invariant across implementations. Specifics of the methods and models for each application are generated from the OpenAPI Generator templates.

NOTE: This class is auto generated by OpenAPI Generator (https://openapi-generator.tech). Ref: https://openapi-generator.tech Do not edit the class manually.

Public fields

`basePath` Base url

`userAgent` Default user agent

`username` Username for HTTP basic authentication

`password` Password for HTTP basic authentication

`timeout` Default timeout in seconds

`retryStatusCodes` vector of status codes to retry

`maxRetryAttempts` maximum number of retries for the status codes

Methods

Public methods:

- `ApiClient$new()`
- `ApiClient$CallApi()`
- `ApiClient$ExecuteWrapped()`
- `ApiClient$Execute()`

- ApiClient$deserialize()
- ApiClient$deserializeObj()
- ApiClient$clone()

Method `new()`**:**

Usage:

```
ApiClient$new(
  basePath = NULL,
  userAgent = NULL,
  defaultHeaders = NULL,
  username = NULL,
  password = NULL,
  apiKeys = NULL,
  accessToken = NULL,
  timeout = NULL,
  retryStatusCodes = NULL,
  maxRetryAttempts = NULL
)
```

Method `CallApi()`**:**

Usage:

```
ApiClient$CallApi(url, method, queryParams, headerParams, body, ...)
```

Method `ExecuteWrapped()`**:**

Usage:

```
ApiClient$ExecuteWrapped(url, method, queryParams, headerParams, body, ...)
```

Method `Execute()`**:**

Usage:

```
ApiClient$Execute(url, method, queryParams, headerParams, body, ...)
```

Method `deserialize()`**:**

Usage:

```
ApiClient$deserialize(resp, returnType, pkgEnv)
```

Method `deserializeObj()`**:**

Usage:

```
ApiClient$deserializeObj(obj, returnType, pkgEnv)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ApiClient$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

ApiResponse *ApiResponse*

Description

ApiResponse Class

Format

An R6Class generator object

Public fields

`content` The deserialized response body.

`response` The raw response from the endpoint.

Methods

Public methods:

- ApiResponse$new()
- ApiResponse$clone()

Method `new()`**:**

Usage:

```
ApiResponse$new(content, response)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ApiResponse$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Array *Array*

Description

Array Class

Format

An R6Class generator object

Public fields

`timestamp` numeric

`queryType` Querytype

`uri` character

Methods

Public methods:

- `Array$new()`
- `Array$toJSON()`
- `Array$fromJSON()`
- `Array$toJSONString()`
- `Array$fromJSONString()`
- `Array$clone()`

Method `new()`:

Usage:

```
Array$new(timestamp, queryType, uri, ...)
```

Method `toJSON()`:

Usage:

```
Array$toJSON()
```

Method `fromJSON()`:

Usage:

```
Array$fromJSON(ArrayJson)
```

Method `toJSONString()`:

Usage:

```
Array$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
Array$fromJSONString(ArrayJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
Array$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ArrayActions *ArrayActions*

Description

ArrayActions Class

Format

An R6Class generator object

Methods

Public methods:

- ArrayActions$new()
- ArrayActions$toJSON()
- ArrayActions$fromJSON()
- ArrayActions$toJSONString()
- ArrayActions$fromJSONString()
- ArrayActions$clone()

Method new():

Usage:

```
ArrayActions$new(...)
```

Method toJSON():

Usage:

```
ArrayActions$toJSON()
```

Method fromJSON():

Usage:

```
ArrayActions$fromJSON(ArrayActionsJson)
```

Method toJSONString():

Usage:

```
ArrayActions$toJSONString()
```

Method fromJSONString():

Usage:

```
ArrayActions$fromJSONString(ArrayActionsJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ArrayActions$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

ArrayActivityLog *ArrayActivityLog*

Description

ArrayActivityLog Class

Format

An R6Class generator object

Public fields

`event_at` character [optional]

`action` ActivityEventType [optional]

`username` character [optional]

`bytes_sent` integer [optional]

`bytes_received` integer [optional]

`array_task_id` character [optional]

`id` character [optional]

`query_ranges` character [optional]

`query_stats` character [optional]

Methods

Public methods:

- `ArrayActivityLog$new()`
- `ArrayActivityLog$toJSON()`
- `ArrayActivityLog$fromJSON()`
- `ArrayActivityLog$toJSONString()`
- `ArrayActivityLog$fromJSONString()`
- `ArrayActivityLog$clone()`

Method `new()`:

Usage:

```
ArrayActivityLog$new(
  event_at = NULL,
  action = NULL,
  username = NULL,
  bytes_sent = NULL,
  bytes_received = NULL,
  array_task_id = NULL,
  id = NULL,
```

```
  query_ranges = NULL,
  query_stats = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
ArrayActivityLog$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
ArrayActivityLog$fromJSON(ArrayActivityLogJson)
```

Method `toJSONString()`**:**

Usage:

```
ArrayActivityLog$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ArrayActivityLog$fromJSONString(ArrayActivityLogJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayActivityLog$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayApi` *Array operations*

Description

tiledbcloud.Array

Format

An `R6Class` generator object

Methods

ArrayActivityLog get array activity logs

@*param* namespace character

- @*param* array character
- @*param* start integer
- @*param* end integer
- @*param* event.types character
- @*param* task.id character
- @*param* has.task.id character
- @*returnType* list(ArrayActivityLog)

- status code : 200 | log of array activity
- return type : array[ArrayActivityLog]
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ArraysBrowserOwnedGet Fetch a list of all arrays that are owned directly by user or user's organizations

@*param* page integer

- @*param* per.page integer
- @*param* search character
- @*param* namespace character
- @*param* orderby character
- @*param* permissions character
- @*param* tag list(character)
- @*param* exclude.tag list(character)
- @*param* file.type list(character)
- @*param* exclude.file.type list(character)
- @*param* file.property list(character)
- @*returnType* ArrayBrowserData

- status code : 200 | Array of array info that are owned directly by user or user's organizations
- return type : ArrayBrowserData
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

ArraysBrowserOwnedSidebarGet Fetch a sidebar for arrays that are owned directly by user or user's organizations

@*returnType* ArrayBrowserSidebar

- status code : 200 | Array of array info that are owned directly by user or user's organizations
- return type : ArrayBrowserSidebar
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ArraysBrowserPublicGet Fetch a list of all arrays that have been shared publically

@*param* page integer
- @*param* per.page integer
- @*param* search character
- @*param* namespace character
- @*param* orderby character
- @*param* permissions character
- @*param* tag list(character)
- @*param* exclude.tag list(character)
- @*param* file.type list(character)
- @*param* exclude.file.type list(character)
- @*param* file.property list(character)
- @*returnType* ArrayBrowserData

- status code : 200 | Array of array info that has been shared publically
- return type : ArrayBrowserData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ArraysBrowserPublicSidebarGet Fetch a sidebar of all arrays that have been shared publically

@*returnType* ArrayBrowserSidebar

- status code : 200 | Array of array info that has been shared publically
- return type : ArrayBrowserSidebar
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ArraysBrowserSharedGet Fetch a list of all arrays that have been shared with the user

@*param* page integer
- @*param* per.page integer
- @*param* search character
- @*param* namespace character
- @*param* orderby character
- @*param* permissions character
- @*param* tag list(character)
- @*param* exclude.tag list(character)
- @*param* file.type list(character)
- @*param* exclude.file.type list(character)
- @*param* file.property list(character)
- @*returnType* ArrayBrowserData

- status code : 200 | Array of array info that has been shared with the user
- return type : ArrayBrowserData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ArraysBrowserSharedSidebarGet Fetch a list of all arrays that have been shared with the user

@*returnType* ArrayBrowserSidebar

- status code : 200 | Array of array info that has been shared with the user
- return type : ArrayBrowserSidebar
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

ArraysNamespaceArrayEndTimestampsGet retrieve a list of timestamps from the array fragment info listing in milliseconds, paginated

@*param* namespace character

- @*param* array character
- @*param* page integer
- @*param* per.page integer
- @*returnType* ArrayEndTimestampData

- status code : 200 | list of timestamps in milliseconds, paginated
- return type : ArrayEndTimestampData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ConsolidateArray consolidate an array at a specified URI

@*param* namespace character

- @*param* array character
- @*param* tiledb.config TileDBConfig
- status code : 204 | array consolidated successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

CreateArray create a array schema at a specified URI registered to a group/project

@*param* namespace character

- @*param* array character
- @*param* content.type character
- @*param* array.schema ArraySchema
- @*param* X_TILEDB_CLOUD_ACCESS_CREDENTIALS_NAME character
- status code : 204 | schema created successfully
- response headers :

- status code : 0 | error response

- return type : Error
- response headers :

DeleteArray delete a array

@*param* namespace character

- @*param* array character
- @*param* content.type character
- status code : 204 | delete array successful
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

DeregisterArray deregister a array

@*param* namespace character

- @*param* array character
- status code : 204 | deregistered array successful
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetActivityLogById get activity log by ID

@*param* namespace character

- @*param* array character
- @*param* id character
- @*returnType* ArrayActivityLog

- status code : 200 | array activity
- return type : ArrayActivityLog
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetAllArrayMetadata get all array metadata user has access to

@*param* public.share character

- @*returnType* list(ArrayInfo)

- status code : 200 | array metadata for all arrays user has access to
- return type : array[ArrayInfo]
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetArray get an ArraySchema using a url encoded uri

@*param* namespace character

- @*param* array character
- @*param* content.type character
- @*returnType* ArraySchema

- status code : 200 | get ArraySchema
- return type : ArraySchema
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetArrayMaxBufferSizes get the max buffer sizes of an array for a subarray

@*param* namespace character

- @*param* array character
- @*param* subarray character
- @*param* content.type character
- @*param* x.payer character
- @*returnType* MaxBufferSizes

- status code : 200 | get the max buffer sizes of an array for a subarray
- return type : MaxBufferSizes
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

GetArrayMetaDataJson get metadata from the array in JSON format

@*param* namespace character

- @*param* array character
- @*param* length integer
- @*param* end.timestamp integer
- status code : 200 | get array metadata
- return type : object
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetArrayMetadata get metadata on an array

@*param* namespace character

- @*param* array character
- @*returnType* ArrayInfo

- status code : 200 | array metadata for an array
- return type : ArrayInfo
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetArrayMetadataCapnp get metadata on an array

@*param* namespace character

- @*param* array character
- @*returnType* ArrayMetadata

- status code : 200 | array metadata for an array
- return type : ArrayMetadata
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

GetArrayNonEmptyDomain get the non empty domain of an array

@*param* namespace character

- @*param* array character
- @*param* content.type character
- @*param* x.payer character
- @*returnType* NonEmptyDomain

- status code : 200 | get the non empty domain of an array
- return type : NonEmptyDomain
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetArrayNonEmptyDomainJson get non-empty domain from the array in json format

@*param* namespace character

- @*param* array character
- status code : 200 | get array non-empty domaim
- return type : object
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetArraySampleData get an sample set of data from the array

@*param* namespace character

- @*param* array character
- @*param* samples numeric
- @*returnType* ArraySample

- status code : 200 | get array sample data
- return type : ArraySample
- response headers :

- status code : 0 | error response

- return type : Error
- response headers :

GetArraySharingPolicies Get all sharing details of the array

@*param* namespace character

- @*param* array character
- @*returnType* list(ArraySharing)

- status code : 200 | List of all specific sharing policies
- return type : array[ArraySharing]
- response headers :

- status code : 404 | Array does not exist or user does not have permissions to view array-sharing policies
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetArraysInNamespace get metadata on all arrays in a namespace

@*param* namespace character

- @*returnType* list(ArrayInfo)

- status code : 200 | array metadata for all arrays in a namespace
- return type : array[ArrayInfo]
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetFragmentEndTimestamp Get fragment end_timestamp on an array, will search for the closest end_timestamp to the timestamp asked

@*param* namespace character

- @*param* array character
- @*param* end.timestamp integer
- status code : 200 | fragment end_timestamp on an array
- return type : integer

- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetLastAccessedArrays

@*returnType* list(LastAccessedArray)

- status code : 200 | gets last accessed arrays
- return type : array[LastAccessedArray]
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

RegisterArray register an array at a specified URI registered to the given namespace

@*param* namespace character

- @*param* array character
- @*param* array.metadata ArrayInfoUpdate
- status code : 204 | schema registered successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ShareArray Share an array with a user

@*param* namespace character

- @*param* array character
- @*param* array.sharing ArraySharing
- status code : 204 | Array shared successfully
- response headers :

- status code : 404 | Array does not exist or user does not have permissions to share array
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

UpdateArrayMetadata update metadata on an array

@*param* namespace character

- @*param* array character
- @*param* array.metadata ArrayInfoUpdate
- status code : 204 | array metadata updated successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

UpdateArrayMetadataCapnp update metadata on an array

@*param* namespace character

- @*param* array character
- @*param* array.metadata.entries ArrayMetadata
- status code : 200 | array metadata updated successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

VacuumArray vacuum an array at a specified URI

@*param* namespace character

- @*param* array character
- @*param* tiledb.config TileDBConfig
- status code : 204 | array vacuumed successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- ArrayApi$new()
- ArrayApi$ArrayActivityLog()
- ArrayApi$ArrayActivityLogWithHttpInfo()
- ArrayApi$ArraysBrowserOwnedGet()
- ArrayApi$ArraysBrowserOwnedGetWithHttpInfo()
- ArrayApi$ArraysBrowserOwnedSidebarGet()
- ArrayApi$ArraysBrowserOwnedSidebarGetWithHttpInfo()
- ArrayApi$ArraysBrowserPublicGet()
- ArrayApi$ArraysBrowserPublicGetWithHttpInfo()
- ArrayApi$ArraysBrowserPublicSidebarGet()
- ArrayApi$ArraysBrowserPublicSidebarGetWithHttpInfo()
- ArrayApi$ArraysBrowserSharedGet()
- ArrayApi$ArraysBrowserSharedGetWithHttpInfo()
- ArrayApi$ArraysBrowserSharedSidebarGet()
- ArrayApi$ArraysBrowserSharedSidebarGetWithHttpInfo()
- ArrayApi$ArraysNamespaceArrayEndTimestampsGet()
- ArrayApi$ArraysNamespaceArrayEndTimestampsGetWithHttpInfo()
- ArrayApi$ConsolidateArray()
- ArrayApi$ConsolidateArrayWithHttpInfo()
- ArrayApi$CreateArray()
- ArrayApi$CreateArrayWithHttpInfo()
- ArrayApi$DeleteArray()
- ArrayApi$DeleteArrayWithHttpInfo()
- ArrayApi$DeregisterArray()
- ArrayApi$DeregisterArrayWithHttpInfo()
- ArrayApi$GetActivityLogById()
- ArrayApi$GetActivityLogByIdWithHttpInfo()
- ArrayApi$GetAllArrayMetadata()
- ArrayApi$GetAllArrayMetadataWithHttpInfo()
- ArrayApi$GetArray()
- ArrayApi$GetArrayWithHttpInfo()
- ArrayApi$GetArrayMaxBufferSizes()
- ArrayApi$GetArrayMaxBufferSizesWithHttpInfo()
- ArrayApi$GetArrayMetaDataJson()
- ArrayApi$GetArrayMetaDataJsonWithHttpInfo()
- ArrayApi$GetArrayMetadata()
- ArrayApi$GetArrayMetadataWithHttpInfo()
- ArrayApi$GetArrayMetadataCapnp()
- ArrayApi$GetArrayMetadataCapnpWithHttpInfo()
- ArrayApi$GetArrayNonEmptyDomain()

- ArrayApi$GetArrayNonEmptyDomainWithHttpInfo()
- ArrayApi$GetArrayNonEmptyDomainJson()
- ArrayApi$GetArrayNonEmptyDomainJsonWithHttpInfo()
- ArrayApi$GetArraySampleData()
- ArrayApi$GetArraySampleDataWithHttpInfo()
- ArrayApi$GetArraySharingPolicies()
- ArrayApi$GetArraySharingPoliciesWithHttpInfo()
- ArrayApi$GetArraysInNamespace()
- ArrayApi$GetArraysInNamespaceWithHttpInfo()
- ArrayApi$GetFragmentEndTimestamp()
- ArrayApi$GetFragmentEndTimestampWithHttpInfo()
- ArrayApi$GetLastAccessedArrays()
- ArrayApi$GetLastAccessedArraysWithHttpInfo()
- ArrayApi$RegisterArray()
- ArrayApi$RegisterArrayWithHttpInfo()
- ArrayApi$ShareArray()
- ArrayApi$ShareArrayWithHttpInfo()
- ArrayApi$UpdateArrayMetadata()
- ArrayApi$UpdateArrayMetadataWithHttpInfo()
- ArrayApi$UpdateArrayMetadataCapnp()
- ArrayApi$UpdateArrayMetadataCapnpWithHttpInfo()
- ArrayApi$VacuumArray()
- ArrayApi$VacuumArrayWithHttpInfo()
- ArrayApi$clone()

Method `new()`**:**

Usage:

```
ArrayApi$new(apiClient)
```

Method `ArrayActivityLog()`**:**

Usage:

```
ArrayApi$ArrayActivityLog(
  namespace,
  array,
  start = NULL,
  end = NULL,
  event.types = NULL,
  task.id = NULL,
  has.task.id = NULL,
  ...
)
```

Method `ArrayActivityLogWithHttpInfo()`**:**

Usage:

```
ArrayApi$ArrayActivityLogWithHttpInfo(
  namespace,
  array,
  start = NULL,
  end = NULL,
  event.types = NULL,
  task.id = NULL,
  has.task.id = NULL,
  ...
)
```

Method `ArraysBrowserOwnedGet()`**:**

Usage:

```
ArrayApi$ArraysBrowserOwnedGet(
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  file.type = NULL,
  exclude.file.type = NULL,
  file.property = NULL,
  ...
)
```

Method `ArraysBrowserOwnedGetWithHttpInfo()`**:**

Usage:

```
ArrayApi$ArraysBrowserOwnedGetWithHttpInfo(
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  file.type = NULL,
  exclude.file.type = NULL,
  file.property = NULL,
  ...
)
```

Method `ArraysBrowserOwnedSidebarGet()`**:**

Usage:

```
ArrayApi$ArraysBrowserOwnedSidebarGet(...)
```

Method `ArraysBrowserOwnedSidebarGetWithHttpInfo()`**:**

Usage:

```
ArrayApi$ArraysBrowserOwnedSidebarGetWithHttpInfo(...)
```

Method `ArraysBrowserPublicGet()`**:**

Usage:

```
ArrayApi$ArraysBrowserPublicGet(
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  file.type = NULL,
  exclude.file.type = NULL,
  file.property = NULL,
  ...
)
```

Method `ArraysBrowserPublicGetWithHttpInfo()`**:**

Usage:

```
ArrayApi$ArraysBrowserPublicGetWithHttpInfo(
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  file.type = NULL,
  exclude.file.type = NULL,
  file.property = NULL,
  ...
)
```

Method `ArraysBrowserPublicSidebarGet()`**:**

Usage:

```
ArrayApi$ArraysBrowserPublicSidebarGet(...)
```

Method `ArraysBrowserPublicSidebarGetWithHttpInfo()`**:**

Usage:

```
ArrayApi$ArraysBrowserPublicSidebarGetWithHttpInfo(...)
```

Method `ArraysBrowserSharedGet()`**:**

Usage:

```
ArrayApi$ArraysBrowserSharedGet(
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  file.type = NULL,
  exclude.file.type = NULL,
  file.property = NULL,
  ...
)
```

Method `ArraysBrowserSharedGetWithHttpInfo()`**:**

Usage:

```
ArrayApi$ArraysBrowserSharedGetWithHttpInfo(
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  file.type = NULL,
  exclude.file.type = NULL,
  file.property = NULL,
  ...
)
```

Method `ArraysBrowserSharedSidebarGet()`**:**

Usage:

```
ArrayApi$ArraysBrowserSharedSidebarGet(...)
```

Method `ArraysBrowserSharedSidebarGetWithHttpInfo()`**:**

Usage:

```
ArrayApi$ArraysBrowserSharedSidebarGetWithHttpInfo(...)
```

Method `ArraysNamespaceArrayEndTimestampsGet()`**:**

Usage:

```
ArrayApi$ArraysNamespaceArrayEndTimestampsGet(
  namespace,
  array,
  page = NULL,
```

```
  per.page = NULL,
  ...
)
```

Method `ArraysNamespaceArrayEndTimestampsGetWithHttpInfo()`**:**

Usage:

```
ArrayApi$ArraysNamespaceArrayEndTimestampsGetWithHttpInfo(
  namespace,
  array,
  page = NULL,
  per.page = NULL,
  ...
)
```

Method `ConsolidateArray()`**:**

Usage:

```
ArrayApi$ConsolidateArray(namespace, array, tiledb.config, ...)
```

Method `ConsolidateArrayWithHttpInfo()`**:**

Usage:

```
ArrayApi$ConsolidateArrayWithHttpInfo(namespace, array, tiledb.config, ...)
```

Method `CreateArray()`**:**

Usage:

```
ArrayApi$CreateArray(
  namespace,
  array,
  content.type,
  array.schema,
  X_TILEDB_CLOUD_ACCESS_CREDENTIALS_NAME = NULL,
  ...
)
```

Method `CreateArrayWithHttpInfo()`**:**

Usage:

```
ArrayApi$CreateArrayWithHttpInfo(
  namespace,
  array,
  content.type,
  array.schema,
  X_TILEDB_CLOUD_ACCESS_CREDENTIALS_NAME = NULL,
  ...
)
```

Method `DeleteArray()`**:**

Usage:

```
ArrayApi$DeleteArray(namespace, array, content.type, ...)
```

Method `DeleteArrayWithHttpInfo()`**:**

Usage:

```
ArrayApi$DeleteArrayWithHttpInfo(namespace, array, content.type, ...)
```

Method `DeregisterArray()`**:**

Usage:

```
ArrayApi$DeregisterArray(namespace, array, ...)
```

Method `DeregisterArrayWithHttpInfo()`**:**

Usage:

```
ArrayApi$DeregisterArrayWithHttpInfo(namespace, array, ...)
```

Method `GetActivityLogById()`**:**

Usage:

```
ArrayApi$GetActivityLogById(namespace, array, id, ...)
```

Method `GetActivityLogByIdWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetActivityLogByIdWithHttpInfo(namespace, array, id, ...)
```

Method `GetAllArrayMetadata()`**:**

Usage:

```
ArrayApi$GetAllArrayMetadata(public.share = NULL, ...)
```

Method `GetAllArrayMetadataWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetAllArrayMetadataWithHttpInfo(public.share = NULL, ...)
```

Method `GetArray()`**:**

Usage:

```
ArrayApi$GetArray(namespace, array, content.type, ...)
```

Method `GetArrayWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetArrayWithHttpInfo(namespace, array, content.type, ...)
```

Method `GetArrayMaxBufferSizes()`**:**

Usage:

```
ArrayApi$GetArrayMaxBufferSizes(
  namespace,
  array,
  subarray,
  content.type,
  x.payer = NULL,
  ...
)
```

Method `GetArrayMaxBufferSizesWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetArrayMaxBufferSizesWithHttpInfo(
  namespace,
  array,
  subarray,
  content.type,
  x.payer = NULL,
  ...
)
```

Method `GetArrayMetaDataJson()`**:**

Usage:

```
ArrayApi$GetArrayMetaDataJson(
  namespace,
  array,
  length = NULL,
  end.timestamp = NULL,
  ...
)
```

Method `GetArrayMetaDataJsonWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetArrayMetaDataJsonWithHttpInfo(
  namespace,
  array,
  length = NULL,
  end.timestamp = NULL,
  ...
)
```

Method `GetArrayMetadata()`**:**

Usage:

```
ArrayApi$GetArrayMetadata(namespace, array, ...)
```

Method `GetArrayMetadataWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetArrayMetadataWithHttpInfo(namespace, array, ...)
```

Method `GetArrayMetadataCapnp()`**:**

Usage:

```
ArrayApi$GetArrayMetadataCapnp(namespace, array, ...)
```

Method `GetArrayMetadataCapnpWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetArrayMetadataCapnpWithHttpInfo(namespace, array, ...)
```

Method `GetArrayNonEmptyDomain()`**:**

Usage:

```
ArrayApi$GetArrayNonEmptyDomain(
  namespace,
  array,
  content.type,
  x.payer = NULL,
  ...
)
```

Method `GetArrayNonEmptyDomainWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetArrayNonEmptyDomainWithHttpInfo(
  namespace,
  array,
  content.type,
  x.payer = NULL,
  ...
)
```

Method `GetArrayNonEmptyDomainJson()`**:**

Usage:

```
ArrayApi$GetArrayNonEmptyDomainJson(namespace, array, ...)
```

Method `GetArrayNonEmptyDomainJsonWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetArrayNonEmptyDomainJsonWithHttpInfo(namespace, array, ...)
```

Method `GetArraySampleData()`**:**

Usage:

```
ArrayApi$GetArraySampleData(namespace, array, samples = 5, ...)
```

Method `GetArraySampleDataWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetArraySampleDataWithHttpInfo(namespace, array, samples = 5, ...)
```

Method `GetArraySharingPolicies()`**:**

Usage:

```
ArrayApi$GetArraySharingPolicies(namespace, array, ...)
```

Method `GetArraySharingPoliciesWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetArraySharingPoliciesWithHttpInfo(namespace, array, ...)
```

Method `GetArraysInNamespace()`**:**

Usage:

```
ArrayApi$GetArraysInNamespace(namespace, ...)
```

Method `GetArraysInNamespaceWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetArraysInNamespaceWithHttpInfo(namespace, ...)
```

Method `GetFragmentEndTimestamp()`**:**

Usage:

```
ArrayApi$GetFragmentEndTimestamp(namespace, array, end.timestamp = NULL, ...)
```

Method `GetFragmentEndTimestampWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetFragmentEndTimestampWithHttpInfo(
  namespace,
  array,
  end.timestamp = NULL,
  ...
)
```

Method `GetLastAccessedArrays()`**:**

Usage:

```
ArrayApi$GetLastAccessedArrays(...)
```

Method `GetLastAccessedArraysWithHttpInfo()`**:**

Usage:

```
ArrayApi$GetLastAccessedArraysWithHttpInfo(...)
```

Method `RegisterArray()`**:**

Usage:

```
ArrayApi$RegisterArray(namespace, array, array.metadata, ...)
```

Method `RegisterArrayWithHttpInfo()`**:**

Usage:

```
ArrayApi$RegisterArrayWithHttpInfo(namespace, array, array.metadata, ...)
```

Method `ShareArray()`**:**

Usage:

```
ArrayApi$ShareArray(namespace, array, array.sharing, ...)
```

Method `ShareArrayWithHttpInfo()`**:**

Usage:

```
ArrayApi$ShareArrayWithHttpInfo(namespace, array, array.sharing, ...)
```

Method `UpdateArrayMetadata()`**:**

Usage:

```
ArrayApi$UpdateArrayMetadata(namespace, array, array.metadata, ...)
```

Method `UpdateArrayMetadataWithHttpInfo()`**:**

Usage:

```
ArrayApi$UpdateArrayMetadataWithHttpInfo(namespace, array, array.metadata, ...)
```

Method `UpdateArrayMetadataCapnp()`**:**

Usage:

```
ArrayApi$UpdateArrayMetadataCapnp(
  namespace,
  array,
  array.metadata.entries,
  ...
)
```

Method `UpdateArrayMetadataCapnpWithHttpInfo()`**:**

Usage:

```
ArrayApi$UpdateArrayMetadataCapnpWithHttpInfo(
  namespace,
  array,
  array.metadata.entries,
  ...
)
```

Method `VacuumArray()`**:**

Usage:

```
ArrayApi$VacuumArray(namespace, array, tiledb.config, ...)
```

Method `VacuumArrayWithHttpInfo()`**:**

Usage:

```
ArrayApi$VacuumArrayWithHttpInfo(namespace, array, tiledb.config, ...)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayApi$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## Not run:
####################  ArrayActivityLog  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.start <- 56 # integer | Start time of window of fetch logs, unix epoch in seconds (default: seven days ago)
var.end <- 56 # integer | End time of window of fetch logs, unix epoch in seconds (default: current utc timestamp)
var.event.types <- 'event.types_example' # character | Event values can be one or more of the following read, write,
```

```
var.task.id <- 'task.id_example' # character | Array task ID To filter activity to
var.has.task.id <- 'has.task.id_example' # character | Excludes activity log results that do not contain an array ta

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ArrayActivityLog(var.namespace, var.array, start=var.start, end=var.end, event.types=var.


####################  ArraysBrowserOwnedGet  ####################

library(tiledbcloud)
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit
var.search <- 'search_example' # character | search string that will look at name, namespace or description fields
var.namespace <- 'namespace_example' # character | namespace
var.orderby <- 'orderby_example' # character | sort by which field valid values include last_accessed, size, name
var.permissions <- 'permissions_example' # character | permissions valid values include read, read_write, write, ad
var.tag <- ['tag_example'] # array[character] | tag to search for, more than one can be included
var.exclude.tag <- ['exclude.tag_example'] # array[character] | tags to exclude matching array in results, more tha
var.file.type <- ['file.type_example'] # array[character] | file_type to search for, more than one can be included
var.exclude.file.type <- ['exclude.file.type_example'] # array[character] | file_type to exclude matching array in
var.file.property <- ['file.property_example'] # array[character] | file_property key-value pair (comma separated,

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ArraysBrowserOwnedGet(page=var.page, per.page=var.per.page, search=var.search, namespace=


####################  ArraysBrowserOwnedSidebarGet  ####################

library(tiledbcloud)

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
```

```
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ArraysBrowserOwnedSidebarGet()


####################  ArraysBrowserPublicGet  ####################

library(tiledbcloud)
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit
var.search <- 'search_example' # character | search string that will look at name, namespace or description fields
var.namespace <- 'namespace_example' # character | namespace
var.orderby <- 'orderby_example' # character | sort by which field valid values include last_accessed, size, name
var.permissions <- 'permissions_example' # character | permissions valid values include read, read_write, write, ad
var.tag <- ['tag_example'] # array[character] | tag to search for, more than one can be included
var.exclude.tag <- ['exclude.tag_example'] # array[character] | tags to exclude matching array in results, more tha
var.file.type <- ['file.type_example'] # array[character] | file_type to search for, more than one can be included
var.exclude.file.type <- ['exclude.file.type_example'] # array[character] | file_type to exclude matching array in
var.file.property <- ['file.property_example'] # array[character] | file_property key-value pair (comma separated,

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ArraysBrowserPublicGet(page=var.page, per.page=var.per.page, search=var.search, namespace


####################  ArraysBrowserPublicSidebarGet  ####################

library(tiledbcloud)

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
```

```
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ArraysBrowserPublicSidebarGet()


####################  ArraysBrowserSharedGet  ####################

library(tiledbcloud)
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit
var.search <- 'search_example' # character | search string that will look at name, namespace or description fields
var.namespace <- 'namespace_example' # character | namespace
var.orderby <- 'orderby_example' # character | sort by which field valid values include last_accessed, size, name
var.permissions <- 'permissions_example' # character | permissions valid values include read, read_write, write, ac
var.tag <- ['tag_example'] # array[character] | tag to search for, more than one can be included
var.exclude.tag <- ['exclude.tag_example'] # array[character] | tags to exclude matching array in results, more tha
var.file.type <- ['file.type_example'] # array[character] | file_type to search for, more than one can be included
var.exclude.file.type <- ['exclude.file.type_example'] # array[character] | file_type to exclude matching array in
var.file.property <- ['file.property_example'] # array[character] | file_property key-value pair (comma separated,

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ArraysBrowserSharedGet(page=var.page, per.page=var.per.page, search=var.search, namespace


####################  ArraysBrowserSharedSidebarGet  ####################

library(tiledbcloud)

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ArraysBrowserSharedSidebarGet()


####################  ArraysNamespaceArrayEndTimestampsGet  ####################
```

```
library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ArraysNamespaceArrayEndTimestampsGet(var.namespace, var.array, page=var.page, per.page=va


####################  ConsolidateArray  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.tiledb.config <- TileDBConfig$new() # TileDBConfig | tiledb configuration

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ConsolidateArray(var.namespace, var.array, var.tiledb.config)


####################  CreateArray  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.content.type <- 'application/json' # character | Content Type of input and return mime
var.array.schema <- ArraySchema$new() # ArraySchema | ArraySchema being created
var.X_TILEDB_CLOUD_ACCESS_CREDENTIALS_NAME <- 'X_TILEDB_CLOUD_ACCESS_CREDENTIALS_NAME_example' # character | Opt

api.instance <- ArrayApi$new()
```

```
#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$CreateArray(var.namespace, var.array, var.content.type, var.array.schema, X_TILEDB_CLOUD_


####################  DeleteArray  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.content.type <- 'application/json' # character | Content Type of input and return mime

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteArray(var.namespace, var.array, var.content.type)


####################  DeregisterArray  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeregisterArray(var.namespace, var.array)
```

```
####################  GetActivityLogById  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.id <- 'id_example' # character | ID of the activity

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetActivityLogById(var.namespace, var.array, var.id)


####################  GetAllArrayMetadata  ####################

library(tiledbcloud)
var.public.share <- 'public.share_example' # character | Public share values can be one of exclude, only

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetAllArrayMetadata(public.share=var.public.share)


####################  GetArray  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.content.type <- 'application/json' # character | Content Type of input and return mime

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
```

```
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetArray(var.namespace, var.array, var.content.type)


####################  GetArrayMaxBufferSizes  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.subarray <- 'subarray_example' # character | CSV string of subarray to get max buffer sizes for
var.content.type <- 'application/json' # character | Content Type of input and return mime
var.x.payer <- 'x.payer_example' # character | Name of organization or user who should be charged for this request

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetArrayMaxBufferSizes(var.namespace, var.array, var.subarray, var.content.type, x.payer=


####################  GetArrayMetaDataJson  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.length <- 56 # integer | (optional) limit character length of returned values
var.end.timestamp <- 56 # integer | Milliseconds since Unix epoch, metadata will use open_at functionality to open a

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetArrayMetaDataJson(var.namespace, var.array, length=var.length, end.timestamp=var.end.t
```

```
####################  GetArrayMetadata  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetArrayMetadata(var.namespace, var.array)


####################  GetArrayMetadataCapnp  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetArrayMetadataCapnp(var.namespace, var.array)


####################  GetArrayNonEmptyDomain  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.content.type <- 'application/json' # character | Content Type of input and return mime
var.x.payer <- 'x.payer_example' # character | Name of organization or user who should be charged for this request

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';
```

```
#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetArrayNonEmptyDomain(var.namespace, var.array, var.content.type, x.payer=var.x.payer)


####################  GetArrayNonEmptyDomainJson  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetArrayNonEmptyDomainJson(var.namespace, var.array)


####################  GetArraySampleData  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.samples <- 5.0 # numeric | Number of sample results to return

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetArraySampleData(var.namespace, var.array, samples=var.samples)


####################  GetArraySharingPolicies  ####################

library(tiledbcloud)
```

```
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetArraySharingPolicies(var.namespace, var.array)


####################  GetArraysInNamespace  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetArraysInNamespace(var.namespace)


####################  GetFragmentEndTimestamp  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.end.timestamp <- 56 # integer | Milliseconds since Unix epoch

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';
```

```
result <- api.instance$GetFragmentEndTimestamp(var.namespace, var.array, end.timestamp=var.end.timestamp)


####################  GetLastAccessedArrays  ####################

library(tiledbcloud)

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetLastAccessedArrays()


####################  RegisterArray  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.array.metadata <- ArrayInfoUpdate$new() # ArrayInfoUpdate | metadata associated with array

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$RegisterArray(var.namespace, var.array, var.array.metadata)


####################  ShareArray  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.array.sharing <- ArraySharing$new() # ArraySharing | Namespace and list of permissions to share with. An empty l

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
```

```
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ShareArray(var.namespace, var.array, var.array.sharing)


####################  UpdateArrayMetadata  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.array.metadata <- ArrayInfoUpdate$new() # ArrayInfoUpdate | array metadata to update

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$UpdateArrayMetadata(var.namespace, var.array, var.array.metadata)


####################  UpdateArrayMetadataCapnp  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.array.metadata.entries <- ArrayMetadata$new() # ArrayMetadata | List of metadata entries

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$UpdateArrayMetadataCapnp(var.namespace, var.array, var.array.metadata.entries)
```

```
####################  VacuumArray  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.tiledb.config <- TileDBConfig$new() # TileDBConfig | tiledb configuration

api.instance <- ArrayApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$VacuumArray(var.namespace, var.array, var.tiledb.config)



## End(Not run)
```

`ArrayBrowserData` *ArrayBrowserData*

Description

ArrayBrowserData Class

Format

An `R6Class` generator object

Public fields

`arrays` list(ArrayInfo) [optional]

`pagination_metadata` PaginationMetadata [optional]

Methods

Public methods:

- `ArrayBrowserData$new()`
- `ArrayBrowserData$toJSON()`
- `ArrayBrowserData$fromJSON()`
- `ArrayBrowserData$toJSONString()`
- `ArrayBrowserData$fromJSONString()`

- ArrayBrowserData$clone()

Method `new()`**:**

Usage:

```
ArrayBrowserData$new(arrays = NULL, pagination_metadata = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
ArrayBrowserData$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
ArrayBrowserData$fromJSON(ArrayBrowserDataJson)
```

Method `toJSONString()`**:**

Usage:

```
ArrayBrowserData$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ArrayBrowserData$fromJSONString(ArrayBrowserDataJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayBrowserData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayBrowserSidebar` *ArrayBrowserSidebar*

Description

ArrayBrowserSidebar Class

Format

An `R6Class` generator object

Public fields

`namespaces` list(character) [optional]

`result_count_for_all` integer [optional]

`result_count_by_namespace` object [optional]

Methods

Public methods:

- ArrayBrowserSidebar$new()
- ArrayBrowserSidebar$toJSON()
- ArrayBrowserSidebar$fromJSON()
- ArrayBrowserSidebar$toJSONString()
- ArrayBrowserSidebar$fromJSONString()
- ArrayBrowserSidebar$clone()

Method new():

Usage:

```
ArrayBrowserSidebar$new(
  namespaces = NULL,
  result_count_for_all = NULL,
  result_count_by_namespace = NULL,
  ...
)
```

Method toJSON():

Usage:

```
ArrayBrowserSidebar$toJSON()
```

Method fromJSON():

Usage:

```
ArrayBrowserSidebar$fromJSON(ArrayBrowserSidebarJson)
```

Method toJSONString():

Usage:

```
ArrayBrowserSidebar$toJSONString()
```

Method fromJSONString():

Usage:

```
ArrayBrowserSidebar$fromJSONString(ArrayBrowserSidebarJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ArrayBrowserSidebar$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`ArrayEndTimestampData` *ArrayEndTimestampData*

Description

ArrayEndTimestampData Class

Format

An `R6Class` generator object

Public fields

`end_timestamps` list(integer) [optional]

`pagination_metadata` PaginationMetadata [optional]

Methods

Public methods:

- `ArrayEndTimestampData$new()`
- `ArrayEndTimestampData$toJSON()`
- `ArrayEndTimestampData$fromJSON()`
- `ArrayEndTimestampData$toJSONString()`
- `ArrayEndTimestampData$fromJSONString()`
- `ArrayEndTimestampData$clone()`

Method `new()`:

Usage:

```
ArrayEndTimestampData$new(
  end_timestamps = NULL,
  pagination_metadata = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
ArrayEndTimestampData$toJSON()
```

Method `fromJSON()`:

Usage:

```
ArrayEndTimestampData$fromJSON(ArrayEndTimestampDataJson)
```

Method `toJSONString()`:

Usage:

```
ArrayEndTimestampData$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ArrayEndTimestampData$fromJSONString(ArrayEndTimestampDataJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayEndTimestampData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayFavorite` *ArrayFavorite*

Description

ArrayFavorite Class

Format

An `R6Class` generator object

Public fields

`array_uuid` character [optional]

`namespace` character [optional]

`name` character [optional]

Methods

Public methods:

- `ArrayFavorite$new()`
- `ArrayFavorite$toJSON()`
- `ArrayFavorite$fromJSON()`
- `ArrayFavorite$toJSONString()`
- `ArrayFavorite$fromJSONString()`
- `ArrayFavorite$clone()`

Method `new()`**:**

Usage:

```
ArrayFavorite$new(array_uuid = NULL, namespace = NULL, name = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
ArrayFavorite$toJSON()
```

Method `fromJSON()`:

Usage:

```
ArrayFavorite$fromJSON(ArrayFavoriteJson)
```

Method `toJSONString()`:

Usage:

```
ArrayFavorite$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
ArrayFavorite$fromJSONString(ArrayFavoriteJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ArrayFavorite$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayFavoritesData` *ArrayFavoritesData*

Description

ArrayFavoritesData Class

Format

An `R6Class` generator object

Public fields

`arrays` list(ArrayInfo) [optional]

`pagination_metadata` PaginationMetadata [optional]

Methods

Public methods:

- `ArrayFavoritesData$new()`
- `ArrayFavoritesData$toJSON()`
- `ArrayFavoritesData$fromJSON()`
- `ArrayFavoritesData$toJSONString()`
- `ArrayFavoritesData$fromJSONString()`
- `ArrayFavoritesData$clone()`

Method `new()`**:**

Usage:

```
ArrayFavoritesData$new(arrays = NULL, pagination_metadata = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
ArrayFavoritesData$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
ArrayFavoritesData$fromJSON(ArrayFavoritesDataJson)
```

Method `toJSONString()`**:**

Usage:

```
ArrayFavoritesData$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ArrayFavoritesData$fromJSONString(ArrayFavoritesDataJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayFavoritesData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayInfo` *ArrayInfo*

Description

ArrayInfo Class

Format

An `R6Class` generator object

Public fields

`id` character [optional]

`file_type` FileType [optional]

`file_properties` named list(character) [optional]

`uri` character [optional]

`namespace` character [optional]

`size` numeric [optional]

`last_accessed` character [optional]

`description` character [optional]

`name` character [optional]

`allowed_actions` list(ArrayActions) [optional]

`pricing` list(Pricing) [optional]

`subscriptions` list(Subscription) [optional]

`logo` character [optional]

`access_credentials_name` character [optional]

`type` character [optional]

`share_count` numeric [optional]

`public_share` character [optional]

`namespace_subscribed` character [optional]

`tiledb_uri` character [optional]

`tags` list(character) [optional]

`license_id` character [optional]

`license_text` character [optional]

`read_only` character [optional]

`is_favorite` character [optional]

Methods

Public methods:

- `ArrayInfo$new()`
- `ArrayInfo$toJSON()`
- `ArrayInfo$fromJSON()`
- `ArrayInfo$toJSONString()`
- `ArrayInfo$fromJSONString()`
- `ArrayInfo$clone()`

Method `new()`:

Usage:

```
ArrayInfo$new(
  id = NULL,
  file_type = NULL,
  file_properties = NULL,
  uri = NULL,
  namespace = NULL,
  size = NULL,
  last_accessed = NULL,
  description = NULL,
  name = NULL,
  allowed_actions = NULL,
  pricing = NULL,
  subscriptions = NULL,
  logo = NULL,
  access_credentials_name = NULL,
  type = NULL,
  share_count = NULL,
  public_share = NULL,
  namespace_subscribed = NULL,
  tiledb_uri = NULL,
  tags = NULL,
  license_id = NULL,
  license_text = NULL,
  read_only = NULL,
  is_favorite = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
ArrayInfo$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
ArrayInfo$fromJSON(ArrayInfoJson)
```

Method `toJSONString()`**:**

Usage:

```
ArrayInfo$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ArrayInfo$fromJSONString(ArrayInfoJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayInfo$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayInfoUpdate` *ArrayInfoUpdate*

Description

ArrayInfoUpdate Class

Format

An `R6Class` generator object

Public fields

`description` character [optional]

`name` character [optional]

`uri` character [optional]

`file_type` FileType [optional]

`file_properties` named list(character) [optional]

`access_credentials_name` character [optional]

`logo` character [optional]

`tags` list(character) [optional]

`license_id` character [optional]

`license_text` character [optional]

`read_only` character [optional]

Methods

Public methods:

- `ArrayInfoUpdate$new()`
- `ArrayInfoUpdate$toJSON()`
- `ArrayInfoUpdate$fromJSON()`
- `ArrayInfoUpdate$toJSONString()`
- `ArrayInfoUpdate$fromJSONString()`
- `ArrayInfoUpdate$clone()`

Method `new()`:

Usage:

```
ArrayInfoUpdate$new(
  description = NULL,
  name = NULL,
  uri = NULL,
  file_type = NULL,
  file_properties = NULL,
```

```
  access_credentials_name = NULL,
  logo = NULL,
  tags = NULL,
  license_id = NULL,
  license_text = NULL,
  read_only = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
ArrayInfoUpdate$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
ArrayInfoUpdate$fromJSON(ArrayInfoUpdateJson)
```

Method `toJSONString()`**:**

Usage:

```
ArrayInfoUpdate$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ArrayInfoUpdate$fromJSONString(ArrayInfoUpdateJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayInfoUpdate$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayMetadata` *ArrayMetadata*

Description

ArrayMetadata Class

Format

An `R6Class` generator object

Public fields

`entries` list(ArrayMetadataEntry) [optional]

Methods

Public methods:

- ArrayMetadata$new()
- ArrayMetadata$toJSON()
- ArrayMetadata$fromJSON()
- ArrayMetadata$toJSONString()
- ArrayMetadata$fromJSONString()
- ArrayMetadata$clone()

Method new():

Usage:

```
ArrayMetadata$new(entries = NULL, ...)
```

Method toJSON():

Usage:

```
ArrayMetadata$toJSON()
```

Method fromJSON():

Usage:

```
ArrayMetadata$fromJSON(ArrayMetadataJson)
```

Method toJSONString():

Usage:

```
ArrayMetadata$toJSONString()
```

Method fromJSONString():

Usage:

```
ArrayMetadata$fromJSONString(ArrayMetadataJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ArrayMetadata$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

ArrayMetadataEntry *ArrayMetadataEntry*

Description

ArrayMetadataEntry Class

Format

An R6Class generator object

Public fields

`key` character [optional]

`type` character [optional]

`valueNum` integer [optional]

`value` list(integer) [optional]

`del` character [optional]

Methods

Public methods:

- `ArrayMetadataEntry$new()`
- `ArrayMetadataEntry$toJSON()`
- `ArrayMetadataEntry$fromJSON()`
- `ArrayMetadataEntry$toJSONString()`
- `ArrayMetadataEntry$fromJSONString()`
- `ArrayMetadataEntry$clone()`

Method `new()`:

Usage:

```
ArrayMetadataEntry$new(
  key = NULL,
  type = NULL,
  valueNum = NULL,
  value = NULL,
  del = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
ArrayMetadataEntry$toJSON()
```

Method `fromJSON()`:

Usage:

```
ArrayMetadataEntry$fromJSON(ArrayMetadataEntryJson)
```

Method `toJSONString()`**:**

Usage:

```
ArrayMetadataEntry$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ArrayMetadataEntry$fromJSONString(ArrayMetadataEntryJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayMetadataEntry$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArraySample` *ArraySample*

Description

ArraySample Class

Format

An `R6Class` generator object

Public fields

`data` object [optional]

Methods

Public methods:

- `ArraySample$new()`
- `ArraySample$toJSON()`
- `ArraySample$fromJSON()`
- `ArraySample$toJSONString()`
- `ArraySample$fromJSONString()`
- `ArraySample$clone()`

Method `new()`**:**

Usage:

```
ArraySample$new(data = NULL, ...)
```

Method `toJSON()`:

Usage:

```
ArraySample$toJSON()
```

Method `fromJSON()`:

Usage:

```
ArraySample$fromJSON(ArraySampleJson)
```

Method `toJSONString()`:

Usage:

```
ArraySample$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
ArraySample$fromJSONString(ArraySampleJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ArraySample$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArraySchema` *ArraySchema*

Description

ArraySchema Class

Format

An `R6Class` generator object

Public fields

`uri` character [optional]

`version` list(integer)

`arrayType` ArrayType

`tileOrder` Layout

`cellOrder` Layout

`capacity` integer

`coordsFilterPipeline` FilterPipeline

`offsetFilterPipeline` FilterPipeline

`domain` Domain

`attributes` list(Attribute)

`allowsDuplicates` character [optional]

Methods

Public methods:

- ArraySchema$new()
- ArraySchema$toJSON()
- ArraySchema$fromJSON()
- ArraySchema$toJSONString()
- ArraySchema$fromJSONString()
- ArraySchema$clone()

Method `new()`**:**

Usage:

```
ArraySchema$new(
  version,
  arrayType,
  tileOrder,
  cellOrder,
  capacity,
  coordsFilterPipeline,
  offsetFilterPipeline,
  domain,
  attributes,
  uri = NULL,
  allowsDuplicates = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
ArraySchema$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
ArraySchema$fromJSON(ArraySchemaJson)
```

Method `toJSONString()`**:**

Usage:

```
ArraySchema$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ArraySchema$fromJSONString(ArraySchemaJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArraySchema$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ArraySharing *ArraySharing*

Description

ArraySharing Class

Format

An R6Class generator object

Public fields

`actions` list(ArrayActions) [optional]

`namespace` character [optional]

`namespace_type` character [optional]

Methods

Public methods:

- `ArraySharing$new()`
- `ArraySharing$toJSON()`
- `ArraySharing$fromJSON()`
- `ArraySharing$toJSONString()`
- `ArraySharing$fromJSONString()`
- `ArraySharing$clone()`

Method `new()`:

Usage:

```
ArraySharing$new(actions = NULL, namespace = NULL, namespace_type = NULL, ...)
```

Method `toJSON()`:

Usage:

```
ArraySharing$toJSON()
```

Method `fromJSON()`:

Usage:

```
ArraySharing$fromJSON(ArraySharingJson)
```

Method `toJSONString()`:

Usage:

```
ArraySharing$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
ArraySharing$fromJSONString(ArraySharingJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArraySharing$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayTask` *ArrayTask*

Description

ArrayTask Class

Format

An `R6Class` generator object

Public fields

`id` character [optional]

`name` character [optional]

`description` character [optional]

`array_metadata` ArrayInfo [optional]

`subarray` DomainArray [optional]

`memory` integer [optional]

`cpu` integer [optional]

`namespace` character [optional]

`status` ArrayTaskStatus [optional]

`start_time` character [optional]

`finish_time` character [optional]

`cost` numeric [optional]

`egress_cost` numeric [optional]

`access_cost` numeric [optional]

`query_type` Querytype [optional]

`udf_code` character [optional]

`udf_language` character [optional]

`sql_query` character [optional]

`type` ArrayTaskType [optional]

`activity` list(ArrayActivityLog) [optional]

`logs` character [optional]

`duration` numeric [optional]

`sql_init_commands` list(character) [optional]

`sql_parameters` list(object) [optional]

`result_format` ResultFormat [optional]

`task_graph_uuid` character [optional]

`client_node_uuid` character [optional]

Methods

Public methods:

- `ArrayTask$new()`
- `ArrayTask$toJSON()`
- `ArrayTask$fromJSON()`
- `ArrayTask$toJSONString()`
- `ArrayTask$fromJSONString()`
- `ArrayTask$clone()`

Method `new()`:

Usage:

```
ArrayTask$new(
  id = NULL,
  name = NULL,
  description = NULL,
  array_metadata = NULL,
  subarray = NULL,
  memory = NULL,
  cpu = NULL,
  namespace = NULL,
  status = NULL,
  start_time = NULL,
  finish_time = NULL,
  cost = NULL,
  egress_cost = NULL,
  access_cost = NULL,
  query_type = NULL,
  udf_code = NULL,
  udf_language = NULL,
  sql_query = NULL,
  type = NULL,
  activity = NULL,
  logs = NULL,
  duration = NULL,
  sql_init_commands = NULL,
```

```
  sql_parameters = NULL,
  result_format = NULL,
  task_graph_uuid = NULL,
  client_node_uuid = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
ArrayTask$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
ArrayTask$fromJSON(ArrayTaskJson)
```

Method `toJSONString()`**:**

Usage:

```
ArrayTask$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ArrayTask$fromJSONString(ArrayTaskJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayTask$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayTaskBrowserSidebar` *ArrayTaskBrowserSidebar*

Description

ArrayTaskBrowserSidebar Class

Format

An `R6Class` generator object

Public fields

`organizations` list(character) [optional]

`result_count_for_all` integer [optional]

`result_count_by_namespace` object [optional]

Methods

Public methods:

- ArrayTaskBrowserSidebar$new()
- ArrayTaskBrowserSidebar$toJSON()
- ArrayTaskBrowserSidebar$fromJSON()
- ArrayTaskBrowserSidebar$toJSONString()
- ArrayTaskBrowserSidebar$fromJSONString()
- ArrayTaskBrowserSidebar$clone()

Method `new()`:

Usage:

```
ArrayTaskBrowserSidebar$new(
  organizations = NULL,
  result_count_for_all = NULL,
  result_count_by_namespace = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
ArrayTaskBrowserSidebar$toJSON()
```

Method `fromJSON()`:

Usage:

```
ArrayTaskBrowserSidebar$fromJSON(ArrayTaskBrowserSidebarJson)
```

Method `toJSONString()`:

Usage:

```
ArrayTaskBrowserSidebar$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
ArrayTaskBrowserSidebar$fromJSONString(ArrayTaskBrowserSidebarJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ArrayTaskBrowserSidebar$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ArrayTaskData *ArrayTaskData*

Description

ArrayTaskData Class

Format

An R6Class generator object

Public fields

`array_tasks` list(ArrayTask) [optional]

`pagination_metadata` PaginationMetadata [optional]

Methods

Public methods:

- `ArrayTaskData$new()`
- `ArrayTaskData$toJSON()`
- `ArrayTaskData$fromJSON()`
- `ArrayTaskData$toJSONString()`
- `ArrayTaskData$fromJSONString()`
- `ArrayTaskData$clone()`

Method `new()`:

Usage:

```
ArrayTaskData$new(array_tasks = NULL, pagination_metadata = NULL, ...)
```

Method `toJSON()`:

Usage:

```
ArrayTaskData$toJSON()
```

Method `fromJSON()`:

Usage:

```
ArrayTaskData$fromJSON(ArrayTaskDataJson)
```

Method `toJSONString()`:

Usage:

```
ArrayTaskData$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
ArrayTaskData$fromJSONString(ArrayTaskDataJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayTaskData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayTaskLog` *ArrayTaskLog*

Description

ArrayTaskLog Class

Format

An `R6Class` generator object

Public fields

`array_task_id` character [optional]

`logs` character [optional]

Methods

Public methods:

- `ArrayTaskLog$new()`
- `ArrayTaskLog$toJSON()`
- `ArrayTaskLog$fromJSON()`
- `ArrayTaskLog$toJSONString()`
- `ArrayTaskLog$fromJSONString()`
- `ArrayTaskLog$clone()`

Method `new()`**:**

Usage:

```
ArrayTaskLog$new(array_task_id = NULL, logs = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
ArrayTaskLog$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
ArrayTaskLog$fromJSON(ArrayTaskLogJson)
```

Method `toJSONString()`:

Usage:

```
ArrayTaskLog$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
ArrayTaskLog$fromJSONString(ArrayTaskLogJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ArrayTaskLog$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayTasksApi` *ArrayTasks operations*

Description

tiledbcloud.ArrayTasks

Format

An `R6Class` generator object

Methods

GetArrayTasksSidebar

@*param* start integer

- @*param* end integer
- @*returnType* ArrayTaskBrowserSidebar

- status code : 200 | sidebar metadata for task definitions for all arrays user has access to
- return type : ArrayTaskBrowserSidebar
- response headers :

- status code : 404 | array tasks not found
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `ArrayTasksApi$new()`
- `ArrayTasksApi$GetArrayTasksSidebar()`
- `ArrayTasksApi$GetArrayTasksSidebarWithHttpInfo()`
- `ArrayTasksApi$clone()`

Method `new()`**:**

Usage:

```
ArrayTasksApi$new(apiClient)
```

Method `GetArrayTasksSidebar()`**:**

Usage:

```
ArrayTasksApi$GetArrayTasksSidebar(start = NULL, end = NULL, ...)
```

Method `GetArrayTasksSidebarWithHttpInfo()`**:**

Usage:

```
ArrayTasksApi$GetArrayTasksSidebarWithHttpInfo(start = NULL, end = NULL, ...)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayTasksApi$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
####################  GetArrayTasksSidebar  ####################

library(tiledbcloud)
var.start <- 56 # integer | Fetch tasks created after this time, unix epoch in seconds, default 7 days ago
var.end <- 56 # integer | Fetch tasks created before this time, unix epoch in seconds, default now

api.instance <- ArrayTasksApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
```

```
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetArrayTasksSidebar(start=var.start, end=var.end)



## End(Not run)
```

ArrayTaskStatus *ArrayTaskStatus*

Description

ArrayTaskStatus Class

Format

An R6Class generator object

Methods

Public methods:

- ArrayTaskStatus$new()
- ArrayTaskStatus$toJSON()
- ArrayTaskStatus$fromJSON()
- ArrayTaskStatus$toJSONString()
- ArrayTaskStatus$fromJSONString()
- ArrayTaskStatus$clone()

Method new():

Usage:

```
ArrayTaskStatus$new(...)
```

Method toJSON():

Usage:

```
ArrayTaskStatus$toJSON()
```

Method fromJSON():

Usage:

```
ArrayTaskStatus$fromJSON(ArrayTaskStatusJson)
```

Method toJSONString():

Usage:

```
ArrayTaskStatus$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ArrayTaskStatus$fromJSONString(ArrayTaskStatusJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayTaskStatus$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayTaskType` *ArrayTaskType*

Description

ArrayTaskType Class

Format

An `R6Class` generator object

Methods

Public methods:

- `ArrayTaskType$new()`
- `ArrayTaskType$toJSON()`
- `ArrayTaskType$fromJSON()`
- `ArrayTaskType$toJSONString()`
- `ArrayTaskType$fromJSONString()`
- `ArrayTaskType$clone()`

Method `new()`**:**

Usage:

```
ArrayTaskType$new(...)
```

Method `toJSON()`**:**

Usage:

```
ArrayTaskType$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
ArrayTaskType$fromJSON(ArrayTaskTypeJson)
```

Method `toJSONString()`**:**

Usage:

```
ArrayTaskType$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ArrayTaskType$fromJSONString(ArrayTaskTypeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayTaskType$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ArrayType` *ArrayType*

Description

ArrayType Class

Format

An `R6Class` generator object

Methods

Public methods:

- `ArrayType$new()`
- `ArrayType$toJSON()`
- `ArrayType$fromJSON()`
- `ArrayType$toJSONString()`
- `ArrayType$fromJSONString()`
- `ArrayType$clone()`

Method `new()`**:**

Usage:

```
ArrayType$new(...)
```

Method `toJSON()`**:**

Usage:

```
ArrayType$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
ArrayType$fromJSON(ArrayTypeJson)
```

Method `toJSONString()`**:**

Usage:

```
ArrayType$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ArrayType$fromJSONString(ArrayTypeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ArrayType$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`array_info` *Show information about an array on TileDB Cloud*

Description

This function shows array information on TileDB Cloud.

Usage

```
array_info(namespace, arrayname)
```

Arguments

`namespace`	Like "TileDB-Inc"
`arrayname`	Like "quickstart_dense"

Details

Nominally you will first call `login`; if not, the results of the last login at `~/.tiledb/cloud.json` will be used.

Value

A list of array properties

See Also

Other manual-layer functions: `compute_sequentially()`, `compute()`, `delayed_args<-()`, `delayed_args()`, `delayed_array_udf()`, `delayed_generic_udf()`, `delayed_sql()`, `delayed()`, `deregister_array()`, `deregister_group()`, `deregister_udf()`, `execute_array_udf()`, `execute_generic_udf()`, `execute_multi_array_ud`
`execute_sql_query()`, `get_udf_info()`, `group_info()`, `list_arrays()`, `list_groups()`, `login()`, `register_array()`, `register_udf()`, `update_udf_info()`, `user_profile()`

Attribute *Attribute*

Description

Attribute Class

Format

An R6Class generator object

Public fields

`name` character

`type` Datatype

`filterPipeline` FilterPipeline

`cellValNum` integer

`nullable` character [optional]

`fillValue` list(integer) [optional]

Methods

Public methods:

- `Attribute$new()`
- `Attribute$toJSON()`
- `Attribute$fromJSON()`
- `Attribute$toJSONString()`
- `Attribute$fromJSONString()`
- `Attribute$clone()`

Method `new()`:

Usage:

```
Attribute$new(
  name,
  type,
  filterPipeline,
  cellValNum,
  nullable = NULL,
  fillValue = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
Attribute$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
Attribute$fromJSON(AttributeJson)
```

Method `toJSONString()`**:**

Usage:

```
Attribute$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Attribute$fromJSONString(AttributeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Attribute$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`AttributeBufferHeader` *AttributeBufferHeader*

Description

AttributeBufferHeader Class

Format

An `R6Class` generator object

Public fields

`name` character

`fixedLenBufferSizeInBytes` integer

`varLenBufferSizeInBytes` integer

Methods

Public methods:

- AttributeBufferHeader$new()
- AttributeBufferHeader$toJSON()
- AttributeBufferHeader$fromJSON()
- AttributeBufferHeader$toJSONString()
- AttributeBufferHeader$fromJSONString()
- AttributeBufferHeader$clone()

Method new():

Usage:

```
AttributeBufferHeader$new(
  name,
  fixedLenBufferSizeInBytes,
  varLenBufferSizeInBytes,
  ...
)
```

Method toJSON():

Usage:

```
AttributeBufferHeader$toJSON()
```

Method fromJSON():

Usage:

```
AttributeBufferHeader$fromJSON(AttributeBufferHeaderJson)
```

Method toJSONString():

Usage:

```
AttributeBufferHeader$toJSONString()
```

Method fromJSONString():

Usage:

```
AttributeBufferHeader$fromJSONString(AttributeBufferHeaderJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
AttributeBufferHeader$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

AttributeBufferSize *AttributeBufferSize*

Description

AttributeBufferSize Class

Format

An R6Class generator object

Public fields

`attribute` character

`offsetBytes` integer

`dataBytes` integer

Methods

Public methods:

- `AttributeBufferSize$new()`
- `AttributeBufferSize$toJSON()`
- `AttributeBufferSize$fromJSON()`
- `AttributeBufferSize$toJSONString()`
- `AttributeBufferSize$fromJSONString()`
- `AttributeBufferSize$clone()`

Method `new()`:

Usage:

```
AttributeBufferSize$new(attribute, offsetBytes, dataBytes, ...)
```

Method `toJSON()`:

Usage:

```
AttributeBufferSize$toJSON()
```

Method `fromJSON()`:

Usage:

```
AttributeBufferSize$fromJSON(AttributeBufferSizeJson)
```

Method `toJSONString()`:

Usage:

```
AttributeBufferSize$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
AttributeBufferSize$fromJSONString(AttributeBufferSizeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
AttributeBufferSize$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`AWSAccessCredentials` *AWSAccessCredentials*

Description

AWSAccessCredentials Class

Format

An `R6Class` generator object

Public fields

`secret_access_key` character [optional]

`access_key_id` character [optional]

`service_role_arn` character [optional]

`name` character [optional]

`default` character [optional]

`buckets` list(character) [optional]

`created_at` character [optional]

`updated_at` character [optional]

Methods

Public methods:

- `AWSAccessCredentials$new()`
- `AWSAccessCredentials$toJSON()`
- `AWSAccessCredentials$fromJSON()`
- `AWSAccessCredentials$toJSONString()`
- `AWSAccessCredentials$fromJSONString()`
- `AWSAccessCredentials$clone()`

Method `new()`**:**

Usage:

```
AWSAccessCredentials$new(
  secret_access_key = NULL,
  access_key_id = NULL,
  service_role_arn = NULL,
  name = NULL,
  default = NULL,
  buckets = NULL,
  created_at = NULL,
  updated_at = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
AWSAccessCredentials$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
AWSAccessCredentials$fromJSON(AWSAccessCredentialsJson)
```

Method `toJSONString()`**:**

Usage:

```
AWSAccessCredentials$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
AWSAccessCredentials$fromJSONString(AWSAccessCredentialsJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
AWSAccessCredentials$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`compute` *Launch a task graph from a given terminal node in the task graph*

Description

The task graph is implicitly defined by various `delayed` objects having others in their argument lists.

Usage

```
compute(
  node,
  timeout_seconds = NULL,
  verbose = FALSE,
  namespace = NULL,
  force_all_local = FALSE
)
```

Arguments

node	The object whose args are being set – nominally, produced by `delayed`, `delayed_generic_udf`, etc.
timeout_seconds	Number of seconds after which to stop waiting for results. Note that in-flight computationsa are not cancelled; this is not supported by the underlying R package we use for concurrency.
verbose	If supplied, show the DAG state at the start and end, along with all node start/end. Also shown are any stdout prints from the individual nodes, but these are only visible once the compute node has completed.
namespace	The namespace to charge for any cloud costs during the execution of the task graph. This can be null only when all nodes have `local`, or when `compute` is called with `force_all_local`.
force_all_local	While individual nodes can be marked with `local=TRUE` to not be executed on TileDB cloud, this flag overrides the default `local=FALSE` for *all* nodes in the task graph.

Value

The value of the computation.

See Also

Other manual-layer functions: `array_info()`, `compute_sequentially()`, `delayed_args<-()`, `delayed_args()`, `delayed_array_udf()`, `delayed_generic_udf()`, `delayed_sql()`, `delayed()`, `deregister_array()`, `deregister_group()`, `deregister_udf()`, `execute_array_udf()`, `execute_generic_udf()`, `execute_multi_array_udf()`, `execute_sql_query()`, `get_udf_info()`, `group_info()`, `list_arrays()`, `list_groups()`, `login()`, `register_array()`, `register_udf()`, `update_udf_info()`, `user_profile()`

---

compute_sequentially *Test/debug entrypoint for local/sequential compute.*

Description

Runs all nodes in a correct dependency ordering, but all within the context of the same process, and all locally. See also the Task Graphs vignette.

Usage

```
compute_sequentially(node)
```

Arguments

node	Nominally, produced by delayed, delayed_generic_udf, etc.

Value

The value of the computation.

See Also

Other manual-layer functions: array_info(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_ud execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

---

configure *Configure TileDB Cloud*

Description

Provide the setup configuration for the TileDB Cloud package.

Usage

```
configure()
```

Details

It considers four different environment variables: TILEDB_REST_TOKEN, TILEDB_REST_HOST, TILEDB_REST_USERNAME, and TILEDB_REST_PASSWORD.

To operate, *either* an API token has to be provided and will be used, *or* the username and password combination will be used to log in with a new session.

Value

A named vector with configuration values is returned.

Datatype *Datatype*

Description

Datatype Class

Format

An `R6Class` generator object

Methods

Public methods:

- `Datatype$new()`
- `Datatype$toJSON()`
- `Datatype$fromJSON()`
- `Datatype$toJSONString()`
- `Datatype$fromJSONString()`
- `Datatype$clone()`

Method `new()`**:**

Usage:

```
Datatype$new(...)
```

Method `toJSON()`**:**

Usage:

```
Datatype$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
Datatype$fromJSON(DatatypeJson)
```

Method `toJSONString()`**:**

Usage:

```
Datatype$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Datatype$fromJSONString(DatatypeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Datatype$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

delayed — *Define a function to be executed within a task graph*

Description

Define a function to be executed within a task graph

Usage

```
delayed(func, args = NULL, name = NULL, namespace = NULL, local = FALSE)
```

Arguments

name	Optional – e.g. a or b. If omitted, it defaults to a UUID.
namespace	If supplied, a namespace to use for executing this particular node. If omitted, a namespace can be applied at your top-level call to compute. If omitted there as well, your logged-in account's default namespace will be used.
local	If true, execute the functions on the local host; if else, execute them as UDFs in TileDB Cloud.

Value

A task-graph node object on which you can later call compute.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_ud execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

delayed_args — *Get arguments for a delayed function, as a list.*

Description

Get arguments for a delayed function, as a list.

Usage

```
delayed_args(node)
```

Arguments

node	The object whose args are being set – nominally, produced by delayed, delayed_generic_udf, etc.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_uc execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

---

delayed_args<- *Set arguments for a delayed function.*

Description

Args can be set when delayed is called, or afterward using this function.

Usage

```
delayed_args(node) <- value
```

Arguments

node	The object whose args are being set – nominally, produced by delayed, delayed_generic_udf, etc.
value	A list of arguments to the delayed function, e.g. list(a,b,c).

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_uc execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

`delayed_array_udf` *Define a single-array UDF to be executed within a task graph*

Description

Define a single-array UDF to be executed within a task graph

Usage

```
delayed_array_udf(
  array,
  udf = NULL,
  registered_udf_name = NULL,
  selectedRanges,
  attrs,
  layout = NULL,
  args = NULL,
  result_format = "native",
  name = NULL,
  namespace = NULL,
  language = "r"
)
```

Arguments

`array`	TileDB URI – see vignette for examples.
`udf`	User-defined function, as in UDF examples. Arguments are specified separately via `args`. One of `udf` and `registered_udf_name` must be non-null.
`registered_udf_name`	Name of a registered UDF, of the form `namespace/udfname`. Arguments are specified separately via `args`. One of `udf` and `registered_udf_name` must be non-null.
`selectedRanges`	As in UDF examples.
`attrs`	As in UDF examples.
`layout`	As in UDF examples.
`result_format`	As in UDF examples.
`name`	A display name for the query
`namespace`	If supplied, a namespace to use for executing this particular node. If omitted, a namespace can be applied at your top-level call to `compute`. If omitted there as well, your logged-in account's default namespace will be used.
`language`	If omitted, defaults to `"r"`. Can be set to `"python"`

Value

The return value from the UDF as an R object.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_ud execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

`delayed_generic_udf` *Define a generic UDF to be executed within a task graph*

Description

Define a generic UDF to be executed within a task graph

Usage

```
delayed_generic_udf(
  udf = NULL,
  registered_udf_name = NULL,
  args = NULL,
  name = NULL,
  namespace = NULL,
  language = "r"
)
```

Arguments

udf	An R function. Arguments are specified separately via args. One of udf and registered_udf_name must be non-null.
registered_udf_name	Name of a registered UDF, of the form namespace/udfname. Arguments are specified separately via args. One of udf and registered_udf_name must be non-null.
name	Optional – e.g. a or b. If omitted, it defaults to a UUID.
namespace	If supplied, a namespace to use for executing this particular node. If omitted, a namespace can be applied at your top-level call to compute. If omitted there as well, your logged-in account's default namespace will be used.
language	If omitted, defaults to "r". Can be set to "python"

Value

The return value from the UDF as an R object.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_uc execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

---

delayed_sql *Define a SQL query function to be executed within a task graph*

Description

Define a SQL query function to be executed within a task graph

Usage

```
delayed_sql(query, name = NULL, namespace = NULL)
```

Arguments

query	SQL query string – see vignette for examples
name	A display name for the query
namespace	If supplied, the TileDB-Cloud namespace to charge the query to. If omitted, a namespace can be applied at your top-level call to compute. If omitted there as well, your logged-in account's default namespace will be used.

Value

A task-graph node object on which you can later call compute. The return value from compute() will be the query result as a dataframe. Note that results will be strings, so numerical results will need to be explicitly cast as such.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_uc execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

deregister_array *Deregister an array from TileDB Cloud*

Description

The underlying storage will not be removed.

Usage

```
deregister_array(namespace = NULL, array_name)
```

Arguments

namespace	Namespace within TileDB cloud to charge. If this is null, the logged-in user's username will be used for the namespace.
array_name	The name to call the array in TileDB Cloud.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_ud execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

deregister_group *De-register a 'Group' object recursively*

Description

This function de-registers a 'Group' object, the 'Group' objects therein as well as any arrays.

Usage

```
deregister_group(
  uri,
  namespace,
  name,
  delete_from_group = TRUE,
  delete_array = FALSE,
  verbose = FALSE
)
```

Arguments

uri	A TileDB + S3 URI
namespace	A character like "TileDB-Inc"
name	A characther "groupABC"
delete_from_group	A logical value, default 'TRUE', whether arrays are removed from the group
delete_array	A logical value, default 'TRUE', whether arrays are deleted too
verbose	A logical value, default 'FALSE', whether operations are verbose or not

Details

Note that 'Group' objects remain on the underlying storage such as S3.

Value

Nothing is returned, the function is invoked for its side-effect

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_ud execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

---

deregister_udf *Deregister a UDF from TileDB Cloud*

Description

Deletes a registered UDF. This removes all sharing and cannot be undone.

Usage

```
deregister_udf(name, namespace)
```

Arguments

name	Name of the UDF in TileDB Cloud, e.g. myudfname.
namespace	Namespace for the UDF in TileDB Cloud, e.g. mynamespace.

Value

No return value.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), execute_array_udf(), execute_generic_udf(), execute_multi_array_udf(), execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

`Dimension` *Dimension*

Description

Dimension Class

Format

An `R6Class` generator object

Public fields

`name` character [optional]

`type` Datatype

`domain` DomainArray

`nullTileExtent` character

`tileExtent` DimensionTileExtent [optional]

`filterPipeline` FilterPipeline [optional]

Methods

Public methods:

- `Dimension$new()`
- `Dimension$toJSON()`
- `Dimension$fromJSON()`
- `Dimension$toJSONString()`
- `Dimension$fromJSONString()`
- `Dimension$clone()`

Method `new()`:

Usage:

```
Dimension$new(
  type,
  domain,
  nullTileExtent,
  name = NULL,
```

```
  tileExtent = NULL,
  filterPipeline = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
Dimension$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
Dimension$fromJSON(DimensionJson)
```

Method `toJSONString()`**:**

Usage:

```
Dimension$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Dimension$fromJSONString(DimensionJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Dimension$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DimensionCoordinate` *DimensionCoordinate*

Description

DimensionCoordinate Class

Format

An `R6Class` generator object

Public fields

`int8` integer [optional]

`uint8` integer [optional]

`int16` integer [optional]

`uint16` integer [optional]

`int32` integer [optional]

`uint32` integer [optional]

`int64` integer [optional]

`uint64` integer [optional]

`float32` numeric [optional]

`float64` numeric [optional]

Methods

Public methods:

- `DimensionCoordinate$new()`
- `DimensionCoordinate$toJSON()`
- `DimensionCoordinate$fromJSON()`
- `DimensionCoordinate$toJSONString()`
- `DimensionCoordinate$fromJSONString()`
- `DimensionCoordinate$clone()`

Method `new()`:

Usage:

```
DimensionCoordinate$new(
  int8 = NULL,
  uint8 = NULL,
  int16 = NULL,
  uint16 = NULL,
  int32 = NULL,
  uint32 = NULL,
  int64 = NULL,
  uint64 = NULL,
  float32 = NULL,
  float64 = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
DimensionCoordinate$toJSON()
```

Method `fromJSON()`:

Usage:

```
DimensionCoordinate$fromJSON(DimensionCoordinateJson)
```

Method `toJSONString()`**:**

Usage:

```
DimensionCoordinate$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
DimensionCoordinate$fromJSONString(DimensionCoordinateJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DimensionCoordinate$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DimensionTileExtent` *DimensionTileExtent*

Description

DimensionTileExtent Class

Format

An `R6Class` generator object

Public fields

`int8` integer [optional]

`uint8` integer [optional]

`int16` integer [optional]

`uint16` integer [optional]

`int32` integer [optional]

`uint32` integer [optional]

`int64` integer [optional]

`uint64` integer [optional]

`float32` integer [optional]

`float64` integer [optional]

Methods

Public methods:

- DimensionTileExtent$new()
- DimensionTileExtent$toJSON()
- DimensionTileExtent$fromJSON()
- DimensionTileExtent$toJSONString()
- DimensionTileExtent$fromJSONString()
- DimensionTileExtent$clone()

Method `new()`**:**

Usage:

```
DimensionTileExtent$new(
  int8 = NULL,
  uint8 = NULL,
  int16 = NULL,
  uint16 = NULL,
  int32 = NULL,
  uint32 = NULL,
  int64 = NULL,
  uint64 = NULL,
  float32 = NULL,
  float64 = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
DimensionTileExtent$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
DimensionTileExtent$fromJSON(DimensionTileExtentJson)
```

Method `toJSONString()`**:**

Usage:

```
DimensionTileExtent$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
DimensionTileExtent$fromJSONString(DimensionTileExtentJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DimensionTileExtent$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Domain *Domain*

Description

Domain Class

Format

An R6Class generator object

Public fields

`type` Datatype

`tileOrder` Layout

`cellOrder` Layout

`dimensions` list(Dimension)

Methods

Public methods:

- `Domain$new()`
- `Domain$toJSON()`
- `Domain$fromJSON()`
- `Domain$toJSONString()`
- `Domain$fromJSONString()`
- `Domain$clone()`

Method `new()`:

Usage:

```
Domain$new(type, tileOrder, cellOrder, dimensions, ...)
```

Method `toJSON()`:

Usage:

```
Domain$toJSON()
```

Method `fromJSON()`:

Usage:

```
Domain$fromJSON(DomainJson)
```

Method `toJSONString()`:

Usage:

```
Domain$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Domain$fromJSONString(DomainJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Domain$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DomainArray` *DomainArray*

Description

DomainArray Class

Format

An `R6Class` generator object

Public fields

`int8` list(integer) [optional]

`uint8` list(integer) [optional]

`int16` list(integer) [optional]

`uint16` list(integer) [optional]

`int32` list(integer) [optional]

`uint32` list(integer) [optional]

`int64` list(integer) [optional]

`uint64` list(integer) [optional]

`float32` list(numeric) [optional]

`float64` list(numeric) [optional]

Methods

Public methods:

- DomainArray$clone()

Method `new()`**:**

Usage:

```
DomainArray$new(
  int8 = NULL,
  uint8 = NULL,
  int16 = NULL,
  uint16 = NULL,
  int32 = NULL,
  uint32 = NULL,
  int64 = NULL,
  uint64 = NULL,
  float32 = NULL,
  float64 = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
DomainArray$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
DomainArray$fromJSON(DomainArrayJson)
```

Method `toJSONString()`**:**

Usage:

```
DomainArray$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
DomainArray$fromJSONString(DomainArrayJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DomainArray$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Error` *Error*

Description

Error Class

Format

An `R6Class` generator object

Public fields

`code` integer [optional]

`message` character [optional]

`request_id` character [optional]

Methods

Public methods:

- `Error$new()`
- `Error$toJSON()`
- `Error$fromJSON()`
- `Error$toJSONString()`
- `Error$fromJSONString()`
- `Error$clone()`

Method `new()`:

Usage:

```
Error$new(code = NULL, message = NULL, request_id = NULL, ...)
```

Method `toJSON()`:

Usage:

```
Error$toJSON()
```

Method `fromJSON()`:

Usage:

```
Error$fromJSON(ErrorJson)
```

Method `toJSONString()`:

Usage:

```
Error$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
Error$fromJSONString(ErrorJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Error$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`execute_array_udf` *Execute a single-array UDF on TileDB Cloud*

Description

This invokes a user-defined function in TileDB Cloud.

Usage

```
execute_array_udf(
  array,
  udf = NULL,
  registered_udf_name = NULL,
  selectedRanges,
  attrs = NULL,
  layout = NULL,
  args = NULL,
  result_format = "native",
  args_format = "native",
  namespace = NULL,
  language = "r",
  resource_class = NULL
)
```

Arguments

`array`	Name of the array, in the form either `tiledb://hello/world` or `hello/world`.
`udf`	An R function which takes a dataframe as argument. Arguments are specified separately via `args`. One of `udf` and `registered_udf_name` must be non-null.
`registered_udf_name`	Name of a registered UDF, of the form `namespace/udfname`. Arguments are specified separately via `args`. One of `udf` and `registered_udf_name` must be non-null.
`selectedRanges`	List of two-column matrices, one matrix per dimension, each matrix being a start-end pair: e.g. `list(cbind(1,10),cbind(1,20))`.

attrs	Optional list of attributes (default: all) for the server-side code to select for UDF execution. Specifying only what your UDF needs is useful for memory-usage control.
layout	One of row-major, col-major, global-order, or unordered,
args	Arguments to the function. If the function takes no arguments, this can be omitted. If you want to call by position, use a list like args=list(123, 456). If you want to call by name, use a named list like args=list(x=123,y=456).
result_format	One of native, json, or arrow. These are used as wire format for returning results from the server to this library, primarily for memory-usage control. UDF return values handed back to your code from this library are converted back to natural R objects.
args_format	One of native, json, or arrow. These are used as wire format for sending arguments to the server. Normally you do not need to specify this. If you're invoking an R UDF, native is used; if you're invoking a registered Python UDF, json is used but you can select arrow if you wish.
namespace	Namespace within TileDB cloud to charge. If this is null, the logged-in user's username will be used for the namespace.
language	If omitted, defaults to "r". Can be set to "python" when executing registered Python UDFs.
resource_class	The resource class to use for the UDF execution. Resource classes define resource limits for memory and CPUs. If this is 'NULL', then the UDF will execute in the standard resource class of the TileDB Cloud provider. This can be set to "large".

Details

Nominally you will first call login; if not, the results of the last login at ~/.tiledb/cloud.json will be used.

All arguments are required.

Value

Return value from the UDF.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_generic_udf(), execute_multi_array_udf execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

`execute_generic_udf` *Execute a generic UDF on TileDB Cloud*

Description

This invokes a user-defined function in TileDB Cloud.

Usage

```
execute_generic_udf(
  udf = NULL,
  registered_udf_name = NULL,
  args = NULL,
  result_format = "native",
  args_format = "native",
  namespace = NULL,
  language = "r",
  resource_class = NULL
)
```

Arguments

`udf`	An R function. Arguments are specified separately via `args`. One of `udf` and `registered_udf_name` must be non-null.
`registered_udf_name`	Name of a registered UDF, of the form `namespace/udfname`. Arguments are specified separately via `args`. One of `udf` and `registered_udf_name` must be non-null.
`args`	Arguments to the function. If the function takes no arguments, this can be omitted. If you want to call by position, use a list like `args=list(123, 456)`. If you want to call by name, use a named list like `args=list(x=123,y=456)`.
`result_format`	One of `native`, `json`, or `arrow`. These are used as wire format for returning results from the server to this library, primarily for memory-usage control. UDF return values handed back to your code from this library are converted back to natural R objects.
`args_format`	One of `native`, `json`, or `arrow`. These are used as wire format for sending arguments to the server. Normally you do not need to specify this. If you're invoking an R UDF, `native` is used; if you're invoking a registered Python UDF, `json` is used but you can select `arrow` if you wish.
`namespace`	Namespace within TileDB cloud to charge. If this is null, the logged-in user's username will be used for the namespace.
`language`	If omitted, defaults to `"r"`. Can be set to `"python"` when executing registered Python UDFs.

`resource_class` The resource class to use for the UDF execution. Resource classes define resource limits for memory and CPUs. If this is 'NULL', then the UDF will execute in the standard resource class of the TileDB Cloud provider. This can be set to `"large"`.

Details

Nominally you will first call `login`; if not, the results of the last login at `~/.tiledb/cloud.json` will be used.

The `udf` and `namespace` arguments are required; the `args` argument is optional.

Value

The R object which is the return value from the UDF.

See Also

Other manual-layer functions: `array_info()`, `compute_sequentially()`, `compute()`, `delayed_args<-()`, `delayed_args()`, `delayed_array_udf()`, `delayed_generic_udf()`, `delayed_sql()`, `delayed()`, `deregister_array()`, `deregister_group()`, `deregister_udf()`, `execute_array_udf()`, `execute_multi_array_udf()`, `execute_sql_query()`, `get_udf_info()`, `group_info()`, `list_arrays()`, `list_groups()`, `login()`, `register_array()`, `register_udf()`, `update_udf_info()`, `user_profile()`

---

execute_multi_array_udf

Execute a multi-array UDF on TileDB Cloud

Description

This invokes a user-defined function in TileDB Cloud.

Usage

```
execute_multi_array_udf(
  array_list,
  udf = NULL,
  registered_udf_name = NULL,
  args = NULL,
  result_format = "native",
  args_format = "native",
  namespace = NULL,
  language = "r",
  resource_class = NULL
)
```

Arguments

array_list	List of UDFArrayDetails objects. Example list element: tiledbcloud::UDFArrayDetails$new(uri=" ranges=QueryRanges$new(layout=Layout$new('row-major'), ranges=list(cbind(1,4),cbind(buffers=list("a"))
udf	An R function which takes dataframes as arguments, one dataframe argument for each element in array_list. Arguments are specified separately via args. One of udf and registered_udf_name must be non-null.
registered_udf_name	Name of a registered UDF, of the form namespace/udfname. Arguments are specified separately via args. One of udf and registered_udf_name must be non-null.
args	Arguments to the function. If the function takes no arguments, this can be omitted. If you want to call by position, use a list like args=list(123, 456). If you want to call by name, use a named list like args=list(x=123,y=456).
result_format	One of native, json, or arrow. These are used as wire format for returning results from the server to this library, primarily for memory-usage control. UDF return values handed back to your code from this library are converted back to natural R objects.
args_format	One of native, json, or arrow. These are used as wire format for sending arguments to the server. Normally you do not need to specify this. If you're invoking an R UDF, native is used; if you're invoking a registered Python UDF, json is used but you can select arrow if you wish.
namespace	Namespace within TileDB cloud to charge. If this is null, the logged-in user's username will be used for the namespace.
language	If omitted, defaults to "r". Can be set to "python" when executing registered Python UDFs.
resource_class	The resource class to use for the UDF execution. Resource classes define resource limits for memory and CPUs. If this is 'NULL', then the UDF will execute in the standard resource class of the TileDB Cloud provider. This can be set to "large".

Details

Nominally you will first call login; if not, the results of the last login at ~/.tiledb/cloud.json will be used.

All arguments are required.

Value

Return value from the UDF.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(),

execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

---

execute_sql_query *Execute a SQL query on TileDB Cloud*

Description

This invokes a user-defined function in TileDB Cloud.

Usage

```
execute_sql_query(query, name = NULL, namespace = NULL)
```

Arguments

query	SQL query as a string.
name	A descriptive name to give the query.
namespace	Namespace within TileDB cloud to charge. If this is null, the logged-in user's username will be used for the namespace.

Details

Nominally you will first call login; if not, the results of the last login at ~/.tiledb/cloud.json will be used.

The udf and namespace arguments are required; the args argument is optional.

Value

The result of the SQL query.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_udf(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

`FavoritesApi` *Favorites operations*

Description

tiledbcloud.Favorites

Format

An `R6Class` generator object

Methods

AddArrayFavorite Add a new array favorite

@*param* namespace character

- @*param* name character
- status code : 204 | Item added to favorites successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

AddMLModelFavorite Add a new ML model favorite

@*param* namespace character

- @*param* name character
- status code : 204 | Item added to favorites successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

AddNotebookFavorite Add a new notebook favorite

@*param* namespace character

- @*param* name character
- status code : 204 | Item added to favorites successfully
- response headers :

- status code : 0 | error response

- return type : Error
- response headers :

AddUDFFavorite Add a new UDF favorite

@*param* namespace character

- @*param* name character
- status code : 204 | Item added to favorites successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

DeleteArrayFavorite Delete specific array favorite

@*param* namespace character

- @*param* name character
- status code : 204 | array favorite item deleted successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

DeleteMLModelFavorite Delete specific ML model favorite

@*param* namespace character

- @*param* name character
- status code : 204 | ML model favorite item deleted successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

DeleteNotebookFavorite Delete specific notebook favorite

@*param* namespace character

- @*param* name character
- status code : 204 | notebook favorite item deleted successfully

- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

DeleteUDFFavorite Delete specific UDF favorite

@*param* namespace character

- @*param* name character
- status code : 204 | UDF favorite item deleted successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetArrayFavorite Fetch array favorite of a specific array

@*param* namespace character

- @*param* name character
- @*returnType* ArrayFavorite

- status code : 200 | OK
- return type : ArrayFavorite
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetMLModelFavorite Fetch ML model favorite of a specific ML model

@*param* namespace character

- @*param* name character
- @*returnType* MLModelFavorite

- status code : 200 | OK
- return type : MLModelFavorite
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetNotebookFavorite Fetch notebook favorite of a specific notebook

@*param* namespace character

- @*param* name character
- @*returnType* NotebookFavorite

- status code : 200 | OK
- return type : NotebookFavorite
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetUDFFavorite Fetch UDF favorite of a specific UDF

@*param* namespace character

- @*param* name character
- @*returnType* UDFFavorite

- status code : 200 | OK
- return type : UDFFavorite
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ListArrayFavorites Fetch a page of array favorites of connected user

@*param* page integer

- @*param* per.page integer
- @*returnType* ArrayFavoritesData

- status code : 200 | Available array favorites are returned
- return type : ArrayFavoritesData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ListArrayFavoritesUUIDs Fetch all favorite array uuids of connected user

@returnType list(ArrayFavorite)

- status code : 200 | Available favorites array uuids are returned
- return type : array[ArrayFavorite]
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ListMLModelFavorites Fetch a page of ML models favorites of connected user

@param page integer
- *@param* per.page integer
- *@returnType* MLModelFavoritesData

- status code : 200 | Available ML models favorites are returned
- return type : MLModelFavoritesData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ListMLModelFavoritesUUIDs Fetch all favorite ML models uuids of connected user

@returnType list(MLModelFavorite)

- status code : 200 | Available favorites ML model uuids are returned
- return type : array[MLModelFavorite]
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

ListNotebookFavorites Fetch a page of notebook favorites of connected user

@*param* is.dashboard character

- @*param* page integer
- @*param* per.page integer
- @*returnType* NotebookFavoritesData

- status code : 200 | Available notebook favorites are returned
- return type : NotebookFavoritesData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ListNotebookFavoritesUUIDs Fetch all favorite notebook uuids of connected user

@*returnType* list(NotebookFavorite)

- status code : 200 | Available favorites notebook uuids are returned
- return type : array[NotebookFavorite]
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ListUDFFavorites Fetch a page of UDF favorites of connected user

@*param* page integer

- @*param* per.page integer
- @*returnType* UDFFavoritesData

- status code : 200 | Available UDF favorites are returned
- return type : UDFFavoritesData
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

ListUDFFavoritesUUIDs Fetch all favorite UDF uuids of connected user
@returnType list(UDFFavorite)

- status code : 200 | Available favorites UDF uuids are returned
- return type : array[UDFFavorite]
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `FavoritesApi$new()`
- `FavoritesApi$AddArrayFavorite()`
- `FavoritesApi$AddArrayFavoriteWithHttpInfo()`
- `FavoritesApi$AddMLModelFavorite()`
- `FavoritesApi$AddMLModelFavoriteWithHttpInfo()`
- `FavoritesApi$AddNotebookFavorite()`
- `FavoritesApi$AddNotebookFavoriteWithHttpInfo()`
- `FavoritesApi$AddUDFFavorite()`
- `FavoritesApi$AddUDFFavoriteWithHttpInfo()`
- `FavoritesApi$DeleteArrayFavorite()`
- `FavoritesApi$DeleteArrayFavoriteWithHttpInfo()`
- `FavoritesApi$DeleteMLModelFavorite()`
- `FavoritesApi$DeleteMLModelFavoriteWithHttpInfo()`
- `FavoritesApi$DeleteNotebookFavorite()`
- `FavoritesApi$DeleteNotebookFavoriteWithHttpInfo()`
- `FavoritesApi$DeleteUDFFavorite()`
- `FavoritesApi$DeleteUDFFavoriteWithHttpInfo()`
- `FavoritesApi$GetArrayFavorite()`
- `FavoritesApi$GetArrayFavoriteWithHttpInfo()`
- `FavoritesApi$GetMLModelFavorite()`
- `FavoritesApi$GetMLModelFavoriteWithHttpInfo()`
- `FavoritesApi$GetNotebookFavorite()`

- FavoritesApi$GetNotebookFavoriteWithHttpInfo()
- FavoritesApi$GetUDFFavorite()
- FavoritesApi$GetUDFFavoriteWithHttpInfo()
- FavoritesApi$ListArrayFavorites()
- FavoritesApi$ListArrayFavoritesWithHttpInfo()
- FavoritesApi$ListArrayFavoritesUUIDs()
- FavoritesApi$ListArrayFavoritesUUIDsWithHttpInfo()
- FavoritesApi$ListMLModelFavorites()
- FavoritesApi$ListMLModelFavoritesWithHttpInfo()
- FavoritesApi$ListMLModelFavoritesUUIDs()
- FavoritesApi$ListMLModelFavoritesUUIDsWithHttpInfo()
- FavoritesApi$ListNotebookFavorites()
- FavoritesApi$ListNotebookFavoritesWithHttpInfo()
- FavoritesApi$ListNotebookFavoritesUUIDs()
- FavoritesApi$ListNotebookFavoritesUUIDsWithHttpInfo()
- FavoritesApi$ListUDFFavorites()
- FavoritesApi$ListUDFFavoritesWithHttpInfo()
- FavoritesApi$ListUDFFavoritesUUIDs()
- FavoritesApi$ListUDFFavoritesUUIDsWithHttpInfo()
- FavoritesApi$clone()

Method `new()`**:**

Usage:

```
FavoritesApi$new(apiClient)
```

Method `AddArrayFavorite()`**:**

Usage:

```
FavoritesApi$AddArrayFavorite(namespace, name, ...)
```

Method `AddArrayFavoriteWithHttpInfo()`**:**

Usage:

```
FavoritesApi$AddArrayFavoriteWithHttpInfo(namespace, name, ...)
```

Method `AddMLModelFavorite()`**:**

Usage:

```
FavoritesApi$AddMLModelFavorite(namespace, name, ...)
```

Method `AddMLModelFavoriteWithHttpInfo()`**:**

Usage:

```
FavoritesApi$AddMLModelFavoriteWithHttpInfo(namespace, name, ...)
```

Method `AddNotebookFavorite()`**:**

Usage:

```
FavoritesApi$AddNotebookFavorite(namespace, name, ...)
```

Method `AddNotebookFavoriteWithHttpInfo()`**:**

Usage:

```
FavoritesApi$AddNotebookFavoriteWithHttpInfo(namespace, name, ...)
```

Method `AddUDFFavorite()`**:**

Usage:

```
FavoritesApi$AddUDFFavorite(namespace, name, ...)
```

Method `AddUDFFavoriteWithHttpInfo()`**:**

Usage:

```
FavoritesApi$AddUDFFavoriteWithHttpInfo(namespace, name, ...)
```

Method `DeleteArrayFavorite()`**:**

Usage:

```
FavoritesApi$DeleteArrayFavorite(namespace, name, ...)
```

Method `DeleteArrayFavoriteWithHttpInfo()`**:**

Usage:

```
FavoritesApi$DeleteArrayFavoriteWithHttpInfo(namespace, name, ...)
```

Method `DeleteMLModelFavorite()`**:**

Usage:

```
FavoritesApi$DeleteMLModelFavorite(namespace, name, ...)
```

Method `DeleteMLModelFavoriteWithHttpInfo()`**:**

Usage:

```
FavoritesApi$DeleteMLModelFavoriteWithHttpInfo(namespace, name, ...)
```

Method `DeleteNotebookFavorite()`**:**

Usage:

```
FavoritesApi$DeleteNotebookFavorite(namespace, name, ...)
```

Method `DeleteNotebookFavoriteWithHttpInfo()`**:**

Usage:

```
FavoritesApi$DeleteNotebookFavoriteWithHttpInfo(namespace, name, ...)
```

Method `DeleteUDFFavorite()`**:**

Usage:

```
FavoritesApi$DeleteUDFFavorite(namespace, name, ...)
```

Method `DeleteUDFFavoriteWithHttpInfo()`**:**

Usage:

```
FavoritesApi$DeleteUDFFavoriteWithHttpInfo(namespace, name, ...)
```

Method `GetArrayFavorite()`**:**

Usage:

```
FavoritesApi$GetArrayFavorite(namespace, name, ...)
```

Method `GetArrayFavoriteWithHttpInfo()`**:**

Usage:

```
FavoritesApi$GetArrayFavoriteWithHttpInfo(namespace, name, ...)
```

Method `GetMLModelFavorite()`**:**

Usage:

```
FavoritesApi$GetMLModelFavorite(namespace, name, ...)
```

Method `GetMLModelFavoriteWithHttpInfo()`**:**

Usage:

```
FavoritesApi$GetMLModelFavoriteWithHttpInfo(namespace, name, ...)
```

Method `GetNotebookFavorite()`**:**

Usage:

```
FavoritesApi$GetNotebookFavorite(namespace, name, ...)
```

Method `GetNotebookFavoriteWithHttpInfo()`**:**

Usage:

```
FavoritesApi$GetNotebookFavoriteWithHttpInfo(namespace, name, ...)
```

Method `GetUDFFavorite()`**:**

Usage:

```
FavoritesApi$GetUDFFavorite(namespace, name, ...)
```

Method `GetUDFFavoriteWithHttpInfo()`**:**

Usage:

```
FavoritesApi$GetUDFFavoriteWithHttpInfo(namespace, name, ...)
```

Method `ListArrayFavorites()`**:**

Usage:

```
FavoritesApi$ListArrayFavorites(page = NULL, per.page = NULL, ...)
```

Method `ListArrayFavoritesWithHttpInfo()`**:**

Usage:

```
FavoritesApi$ListArrayFavoritesWithHttpInfo(page = NULL, per.page = NULL, ...)
```

Method `ListArrayFavoritesUUIDs()`**:**

Usage:

```
FavoritesApi$ListArrayFavoritesUUIDs(...)
```

Method `ListArrayFavoritesUUIDsWithHttpInfo()`**:**

Usage:

```
FavoritesApi$ListArrayFavoritesUUIDsWithHttpInfo(...)
```

Method `ListMLModelFavorites()`**:**

Usage:

```
FavoritesApi$ListMLModelFavorites(page = NULL, per.page = NULL, ...)
```

Method `ListMLModelFavoritesWithHttpInfo()`**:**

Usage:

```
FavoritesApi$ListMLModelFavoritesWithHttpInfo(
  page = NULL,
  per.page = NULL,
  ...
)
```

Method `ListMLModelFavoritesUUIDs()`**:**

Usage:

```
FavoritesApi$ListMLModelFavoritesUUIDs(...)
```

Method `ListMLModelFavoritesUUIDsWithHttpInfo()`**:**

Usage:

```
FavoritesApi$ListMLModelFavoritesUUIDsWithHttpInfo(...)
```

Method `ListNotebookFavorites()`**:**

Usage:

```
FavoritesApi$ListNotebookFavorites(
  is.dashboard = NULL,
  page = NULL,
  per.page = NULL,
  ...
)
```

Method `ListNotebookFavoritesWithHttpInfo()`**:**

Usage:

```
FavoritesApi$ListNotebookFavoritesWithHttpInfo(
  is.dashboard = NULL,
  page = NULL,
  per.page = NULL,
  ...
)
```

Method `ListNotebookFavoritesUUIDs()`**:**

Usage:

```
FavoritesApi$ListNotebookFavoritesUUIDs(...)
```

Method `ListNotebookFavoritesUUIDsWithHttpInfo()`**:**

Usage:

```
FavoritesApi$ListNotebookFavoritesUUIDsWithHttpInfo(...)
```

Method `ListUDFFavorites()`**:**

Usage:

```
FavoritesApi$ListUDFFavorites(page = NULL, per.page = NULL, ...)
```

Method `ListUDFFavoritesWithHttpInfo()`**:**

Usage:

```
FavoritesApi$ListUDFFavoritesWithHttpInfo(page = NULL, per.page = NULL, ...)
```

Method `ListUDFFavoritesUUIDs()`**:**

Usage:

```
FavoritesApi$ListUDFFavoritesUUIDs(...)
```

Method `ListUDFFavoritesUUIDsWithHttpInfo()`**:**

Usage:

```
FavoritesApi$ListUDFFavoritesUUIDsWithHttpInfo(...)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FavoritesApi$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## Not run:
####################  AddArrayFavorite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the array
var.name <- 'name_example' # character | The name of the array

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$AddArrayFavorite(var.namespace, var.name)


####################  AddMLModelFavorite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the ML model
var.name <- 'name_example' # character | The name of the ML model
```

```
api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$AddMLModelFavorite(var.namespace, var.name)


####################  AddNotebookFavorite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the notebook
var.name <- 'name_example' # character | The name of the notebook

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$AddNotebookFavorite(var.namespace, var.name)


####################  AddUDFFavorite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the UDF
var.name <- 'name_example' # character | The name of the UDF

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$AddUDFFavorite(var.namespace, var.name)
```

```
####################  DeleteArrayFavorite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the array
var.name <- 'name_example' # character | The name of the array

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteArrayFavorite(var.namespace, var.name)


####################  DeleteMLModelFavorite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the ML model
var.name <- 'name_example' # character | The name of the ML model

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteMLModelFavorite(var.namespace, var.name)


####################  DeleteNotebookFavorite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the notebook
var.name <- 'name_example' # character | The name of the notebook

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';
```

```
#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteNotebookFavorite(var.namespace, var.name)


####################  DeleteUDFFavorite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the UDF
var.name <- 'name_example' # character | The name of the UDF

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteUDFFavorite(var.namespace, var.name)


####################  GetArrayFavorite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the array
var.name <- 'name_example' # character | The name of the array

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetArrayFavorite(var.namespace, var.name)


####################  GetMLModelFavorite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the ML model
```

```
var.name <- 'name_example' # character | The name of the ML model

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetMLModelFavorite(var.namespace, var.name)


####################  GetNotebookFavorite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the notebook
var.name <- 'name_example' # character | The name of the notebook

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetNotebookFavorite(var.namespace, var.name)


####################  GetUDFFavorite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the UDF
var.name <- 'name_example' # character | The name of the UDF

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';
```

```
result <- api.instance$GetUDFFavorite(var.namespace, var.name)


####################  ListArrayFavorites  ####################

library(tiledbcloud)
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ListArrayFavorites(page=var.page, per.page=var.per.page)


####################  ListArrayFavoritesUUIDs  ####################

library(tiledbcloud)

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ListArrayFavoritesUUIDs()


####################  ListMLModelFavorites  ####################

library(tiledbcloud)
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
```

```
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ListMLModelFavorites(page=var.page, per.page=var.per.page)


####################  ListMLModelFavoritesUUIDs  ####################

library(tiledbcloud)

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ListMLModelFavoritesUUIDs()


####################  ListNotebookFavorites  ####################

library(tiledbcloud)
var.is.dashboard <- 'is.dashboard_example' # character | return only dashboards
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ListNotebookFavorites(is.dashboard=var.is.dashboard, page=var.page, per.page=var.per.page


####################  ListNotebookFavoritesUUIDs  ####################

library(tiledbcloud)

api.instance <- FavoritesApi$new()
```

```
#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ListNotebookFavoritesUUIDs()


####################  ListUDFFavorites  ####################

library(tiledbcloud)
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ListUDFFavorites(page=var.page, per.page=var.per.page)


####################  ListUDFFavoritesUUIDs  ####################

library(tiledbcloud)

api.instance <- FavoritesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ListUDFFavoritesUUIDs()



## End(Not run)
```

FileCreate *FileCreate*

Description

FileCreate Class

Format

An R6Class generator object

Public fields

`input_uri` character [optional]

`output_uri` character [optional]

`name` character [optional]

Methods

Public methods:

- FileCreate$new()
- FileCreate$toJSON()
- FileCreate$fromJSON()
- FileCreate$toJSONString()
- FileCreate$fromJSONString()
- FileCreate$clone()

Method `new()`:

Usage:

```
FileCreate$new(input_uri = NULL, output_uri = NULL, name = NULL, ...)
```

Method `toJSON()`:

Usage:

```
FileCreate$toJSON()
```

Method `fromJSON()`:

Usage:

```
FileCreate$fromJSON(FileCreateJson)
```

Method `toJSONString()`:

Usage:

```
FileCreate$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
FileCreate$fromJSONString(FileCreateJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FileCreate$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`FileCreated` *FileCreated*

Description

FileCreated Class

Format

An `R6Class` generator object

Public fields

`output_uri` character [optional]

`file_name` character [optional]

Methods

Public methods:

- `FileCreated$new()`
- `FileCreated$toJSON()`
- `FileCreated$fromJSON()`
- `FileCreated$toJSONString()`
- `FileCreated$fromJSONString()`
- `FileCreated$clone()`

Method `new()`**:**

Usage:

```
FileCreated$new(output_uri = NULL, file_name = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
FileCreated$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
FileCreated$fromJSON(FileCreatedJson)
```

Method `toJSONString()`:

Usage:

```
FileCreated$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
FileCreated$fromJSONString(FileCreatedJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
FileCreated$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`FileExport` *FileExport*

Description

FileExport Class

Format

An `R6Class` generator object

Public fields

`output_uri` character [optional]

Methods

Public methods:

- `FileExport$new()`
- `FileExport$toJSON()`
- `FileExport$fromJSON()`
- `FileExport$toJSONString()`
- `FileExport$fromJSONString()`
- `FileExport$clone()`

Method `new()`:

Usage:

```
FileExport$new(output_uri = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
FileExport$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
FileExport$fromJSON(FileExportJson)
```

Method `toJSONString()`**:**

Usage:

```
FileExport$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
FileExport$fromJSONString(FileExportJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FileExport$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`FileExported` *FileExported*

Description

FileExported Class

Format

An `R6Class` generator object

Public fields

`output_uri` character [optional]

Methods

Public methods:

- FileExported$new()
- FileExported$toJSON()
- FileExported$fromJSON()
- FileExported$toJSONString()
- FileExported$fromJSONString()
- FileExported$clone()

Method new():

Usage:

```
FileExported$new(output_uri = NULL, ...)
```

Method toJSON():

Usage:

```
FileExported$toJSON()
```

Method fromJSON():

Usage:

```
FileExported$fromJSON(FileExportedJson)
```

Method toJSONString():

Usage:

```
FileExported$toJSONString()
```

Method fromJSONString():

Usage:

```
FileExported$fromJSONString(FileExportedJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
FileExported$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

FilePropertyName *FilePropertyName*

Description

FilePropertyName Class

Format

An R6Class generator object

Methods

Public methods:

- FilePropertyName$new()
- FilePropertyName$toJSON()
- FilePropertyName$fromJSON()
- FilePropertyName$toJSONString()
- FilePropertyName$fromJSONString()
- FilePropertyName$clone()

Method `new()`**:**

Usage:

```
FilePropertyName$new(...)
```

Method `toJSON()`**:**

Usage:

```
FilePropertyName$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
FilePropertyName$fromJSON(FilePropertyNameJson)
```

Method `toJSONString()`**:**

Usage:

```
FilePropertyName$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
FilePropertyName$fromJSONString(FilePropertyNameJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FilePropertyName$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`FilesApi` *Files operations*

Description

tiledbcloud.Files

Format

An `R6Class` generator object

Methods

HandleCreateFile Create a tiledb file at the specified location

@*param* namespace character

- @*param* file.create FileCreate
- @*param* X_TILEDB_CLOUD_ACCESS_CREDENTIALS_NAME character
- @*returnType* FileCreated

- status code : 201 | File created
- return type : FileCreated
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

HandleExportFile Export a TileDB File back to its original file format

@*param* namespace character

- @*param* file character
- @*param* file.export FileExport
- @*returnType* FileExported

- status code : 201 | File exported
- return type : FileExported
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `FilesApi$new()`
- `FilesApi$HandleCreateFile()`
- `FilesApi$HandleCreateFileWithHttpInfo()`
- `FilesApi$HandleExportFile()`
- `FilesApi$HandleExportFileWithHttpInfo()`
- `FilesApi$clone()`

Method `new()`**:**

Usage:

```
FilesApi$new(apiClient)
```

Method `HandleCreateFile()`**:**

Usage:

```
FilesApi$HandleCreateFile(
  namespace,
  file.create,
  X_TILEDB_CLOUD_ACCESS_CREDENTIALS_NAME = NULL,
  ...
)
```

Method `HandleCreateFileWithHttpInfo()`**:**

Usage:

```
FilesApi$HandleCreateFileWithHttpInfo(
  namespace,
  file.create,
  X_TILEDB_CLOUD_ACCESS_CREDENTIALS_NAME = NULL,
  ...
)
```

Method `HandleExportFile()`**:**

Usage:

```
FilesApi$HandleExportFile(namespace, file, file.export, ...)
```

Method `HandleExportFileWithHttpInfo()`**:**

Usage:

```
FilesApi$HandleExportFileWithHttpInfo(namespace, file, file.export, ...)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FilesApi$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
####################  HandleCreateFile  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the file
var.file.create <- FileCreate$new() # FileCreate | Input/Output information to create a new TileDB file
var.X_TILEDB_CLOUD_ACCESS_CREDENTIALS_NAME <- 'X_TILEDB_CLOUD_ACCESS_CREDENTIALS_NAME_example' # character | Opt

api.instance <- FilesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$HandleCreateFile(var.namespace, var.file.create, X_TILEDB_CLOUD_ACCESS_CREDENTIALS_NAME=v


####################  HandleExportFile  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the file
var.file <- 'file_example' # character | The file identifier
var.file.export <- FileExport$new() # FileExport | Export configuration information

api.instance <- FilesApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$HandleExportFile(var.namespace, var.file, var.file.export)



## End(Not run)
```

FileType *FileType*

Description

FileType Class

Format

An `R6Class` generator object

Methods

Public methods:

- `FileType$new()`
- `FileType$toJSON()`
- `FileType$fromJSON()`
- `FileType$toJSONString()`
- `FileType$fromJSONString()`
- `FileType$clone()`

Method `new()`**:**

Usage:

```
FileType$new(...)
```

Method `toJSON()`**:**

Usage:

```
FileType$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
FileType$fromJSON(FileTypeJson)
```

Method `toJSONString()`**:**

Usage:

```
FileType$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
FileType$fromJSONString(FileTypeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FileType$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`Filter` *Filter*

Description

Filter Class

Format

An `R6Class` generator object

Public fields

`type` FilterType

`data` FilterData [optional]

Methods

Public methods:

- `Filter$new()`
- `Filter$toJSON()`
- `Filter$fromJSON()`
- `Filter$toJSONString()`
- `Filter$fromJSONString()`
- `Filter$clone()`

Method `new()`:

Usage:

```
Filter$new(type, data = NULL, ...)
```

Method `toJSON()`:

Usage:

```
Filter$toJSON()
```

Method `fromJSON()`:

Usage:

```
Filter$fromJSON(FilterJson)
```

Method `toJSONString()`:

Usage:

```
Filter$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
Filter$fromJSONString(FilterJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Filter$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`FilterData` *FilterData*

Description

FilterData Class

Format

An `R6Class` generator object

Public fields

`int8` integer [optional]

`uint8` integer [optional]

`int16` integer [optional]

`uint16` integer [optional]

`int32` integer [optional]

`uint32` integer [optional]

`int64` integer [optional]

`uint64` integer [optional]

`float32` integer [optional]

`float64` integer [optional]

Methods

Public methods:

- `FilterData$new()`
- `FilterData$toJSON()`
- `FilterData$fromJSON()`
- `FilterData$toJSONString()`
- `FilterData$fromJSONString()`
- `FilterData$clone()`

Method `new()`**:**

Usage:

```
FilterData$new(
  int8 = NULL,
  uint8 = NULL,
  int16 = NULL,
  uint16 = NULL,
  int32 = NULL,
  uint32 = NULL,
  int64 = NULL,
  uint64 = NULL,
  float32 = NULL,
  float64 = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
FilterData$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
FilterData$fromJSON(FilterDataJson)
```

Method `toJSONString()`**:**

Usage:

```
FilterData$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
FilterData$fromJSONString(FilterDataJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FilterData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`FilterOption` *FilterOption*

Description

FilterOption Class

Format

An `R6Class` generator object

Methods

Public methods:

- FilterOption$new()
- FilterOption$toJSON()
- FilterOption$fromJSON()
- FilterOption$toJSONString()
- FilterOption$fromJSONString()
- FilterOption$clone()

Method new():

Usage:

```
FilterOption$new(...)
```

Method toJSON():

Usage:

```
FilterOption$toJSON()
```

Method fromJSON():

Usage:

```
FilterOption$fromJSON(FilterOptionJson)
```

Method toJSONString():

Usage:

```
FilterOption$toJSONString()
```

Method fromJSONString():

Usage:

```
FilterOption$fromJSONString(FilterOptionJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
FilterOption$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

FilterPipeline *FilterPipeline*

Description

FilterPipeline Class

Format

An R6Class generator object

Public fields

`filters` list(Filter) [optional]

Methods

Public methods:

- `FilterPipeline$new()`
- `FilterPipeline$toJSON()`
- `FilterPipeline$fromJSON()`
- `FilterPipeline$toJSONString()`
- `FilterPipeline$fromJSONString()`
- `FilterPipeline$clone()`

Method `new()`:

Usage:

```
FilterPipeline$new(filters = NULL, ...)
```

Method `toJSON()`:

Usage:

```
FilterPipeline$toJSON()
```

Method `fromJSON()`:

Usage:

```
FilterPipeline$fromJSON(FilterPipelineJson)
```

Method `toJSONString()`:

Usage:

```
FilterPipeline$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
FilterPipeline$fromJSONString(FilterPipelineJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FilterPipeline$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`FilterType` *FilterType*

Description

FilterType Class

Format

An `R6Class` generator object

Methods

Public methods:

- `FilterType$new()`
- `FilterType$toJSON()`
- `FilterType$fromJSON()`
- `FilterType$toJSONString()`
- `FilterType$fromJSONString()`
- `FilterType$clone()`

Method `new()`**:**

Usage:

```
FilterType$new(...)
```

Method `toJSON()`**:**

Usage:

```
FilterType$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
FilterType$fromJSON(FilterTypeJson)
```

Method `toJSONString()`**:**

Usage:

```
FilterType$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
FilterType$fromJSONString(FilterTypeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FilterType$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GenericUDF` *GenericUDF*

Description

GenericUDF Class

Format

An `R6Class` generator object

Public fields

`udf_info_name` character [optional]

`language` UDFLanguage [optional]

`version` character [optional]

`image_name` character [optional]

`resource_class` character [optional]

`exec` character [optional]

`exec_raw` character [optional]

`argument` character [optional]

`stored_param_uuids` list(character) [optional]

`result_format` ResultFormat [optional]

`task_name` character [optional]

`store_results` character [optional]

`timeout` integer [optional]

`dont_download_results` character [optional]

`task_graph_uuid` character [optional]

`client_node_uuid` character [optional]

Methods

Public methods:

- GenericUDF$new()
- GenericUDF$toJSON()
- GenericUDF$fromJSON()
- GenericUDF$toJSONString()
- GenericUDF$fromJSONString()
- GenericUDF$clone()

Method `new()`:

Usage:

```
GenericUDF$new(
  udf_info_name = NULL,
  language = NULL,
  version = NULL,
  image_name = NULL,
  resource_class = NULL,
  exec = NULL,
  exec_raw = NULL,
  argument = NULL,
  stored_param_uuids = NULL,
  result_format = NULL,
  task_name = NULL,
  store_results = NULL,
  timeout = NULL,
  dont_download_results = NULL,
  task_graph_uuid = NULL,
  client_node_uuid = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
GenericUDF$toJSON()
```

Method `fromJSON()`:

Usage:

```
GenericUDF$fromJSON(GenericUDFJson)
```

Method `toJSONString()`:

Usage:

```
GenericUDF$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
GenericUDF$fromJSONString(GenericUDFJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GenericUDF$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`get_api_client_instance` *Access cached API-client object*

Description

This is a package-internal function.

Usage

```
get_api_client_instance()
```

Details

It returns the cached `ApiClient` object, or stops. The `ApiClient` instance is constructor input to `UserApi`, `ArrayApi`, etc. as generated by OpenAPI. Note that with in the OpenAPI autogen code, there is the `ApiClient` instance which is used as constructor input to `UserApi`, `ArrayApi`, `UdfAPI`, etc.

Value

The cached `ApiClient` object, or stops.

`get_udf_info` *Get information about a UDF on TileDB Cloud*

Description

Reads back information for a specified user-defined function on TileDB Cloud. Note that `version`, `image_name`, `exec`, and `exec_raw` are writable via `register_udf` but are not read back by this function.

Usage

```
get_udf_info(name, namespace)
```

Arguments

`name`	Name of the UDF in TileDB Cloud, e.g. `myudfname`.
`namespace`	Namespace for the UDF in TileDB Cloud, e.g. `mynamespace`.

Value

List of key-value pairs of UDF information.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_udf(), execute_sql_query(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

Group *Group*

Description

Group Class

Format

An R6Class generator object

Public fields

`id` character [optional]

`namespace` character [optional]

`name` character [optional]

`description` character [optional]

Methods

Public methods:

- Group$new()
- Group$toJSON()
- Group$fromJSON()
- Group$toJSONString()
- Group$fromJSONString()
- Group$clone()

Method `new()`:

Usage:

```
Group$new(id = NULL, namespace = NULL, name = NULL, description = NULL, ...)
```

Method `toJSON()`:

Usage:

```
Group$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
Group$fromJSON(GroupJson)
```

Method `toJSONString()`**:**

Usage:

```
Group$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Group$fromJSONString(GroupJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Group$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GroupActions` *GroupActions*

Description

GroupActions Class

Format

An `R6Class` generator object

Methods

Public methods:

- `GroupActions$new()`
- `GroupActions$toJSON()`
- `GroupActions$fromJSON()`
- `GroupActions$toJSONString()`
- `GroupActions$fromJSONString()`
- `GroupActions$clone()`

Method `new()`**:**

Usage:

```
GroupActions$new(...)
```

Method `toJSON()`:

Usage:

```
GroupActions$toJSON()
```

Method `fromJSON()`:

Usage:

```
GroupActions$fromJSON(GroupActionsJson)
```

Method `toJSONString()`:

Usage:

```
GroupActions$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
GroupActions$fromJSONString(GroupActionsJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
GroupActions$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GroupBrowserData` *GroupBrowserData*

Description

GroupBrowserData Class

Format

An `R6Class` generator object

Public fields

`groups` list(GroupInfo) [optional]

`pagination_metadata` PaginationMetadata [optional]

Methods

Public methods:

- GroupBrowserData$new()
- GroupBrowserData$toJSON()
- GroupBrowserData$fromJSON()
- GroupBrowserData$toJSONString()
- GroupBrowserData$fromJSONString()
- GroupBrowserData$clone()

Method `new()`:

Usage:

```
GroupBrowserData$new(groups = NULL, pagination_metadata = NULL, ...)
```

Method `toJSON()`:

Usage:

```
GroupBrowserData$toJSON()
```

Method `fromJSON()`:

Usage:

```
GroupBrowserData$fromJSON(GroupBrowserDataJson)
```

Method `toJSONString()`:

Usage:

```
GroupBrowserData$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
GroupBrowserData$fromJSONString(GroupBrowserDataJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
GroupBrowserData$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

GroupBrowserFilterData *GroupBrowserFilterData*

Description

GroupBrowserFilterData Class

Format

An `R6Class` generator object

Public fields

`namespaces` list(character) [optional]

Methods

Public methods:

- `GroupBrowserFilterData$new()`
- `GroupBrowserFilterData$toJSON()`
- `GroupBrowserFilterData$fromJSON()`
- `GroupBrowserFilterData$toJSONString()`
- `GroupBrowserFilterData$fromJSONString()`
- `GroupBrowserFilterData$clone()`

Method `new()`:

Usage:

```
GroupBrowserFilterData$new(namespaces = NULL, ...)
```

Method `toJSON()`:

Usage:

```
GroupBrowserFilterData$toJSON()
```

Method `fromJSON()`:

Usage:

```
GroupBrowserFilterData$fromJSON(GroupBrowserFilterDataJson)
```

Method `toJSONString()`:

Usage:

```
GroupBrowserFilterData$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
GroupBrowserFilterData$fromJSONString(GroupBrowserFilterDataJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupBrowserFilterData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GroupChanges` *GroupChanges*

Description

GroupChanges Class

Format

An `R6Class` generator object

Public fields

`add` list(GroupMember) [optional]

`remove` list(GroupMember) [optional]

Methods

Public methods:

- `GroupChanges$new()`
- `GroupChanges$toJSON()`
- `GroupChanges$fromJSON()`
- `GroupChanges$toJSONString()`
- `GroupChanges$fromJSONString()`
- `GroupChanges$clone()`

Method `new()`**:**

Usage:

```
GroupChanges$new(add = NULL, remove = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
GroupChanges$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
GroupChanges$fromJSON(GroupChangesJson)
```

Method `toJSONString()`:

Usage:

```
GroupChanges$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
GroupChanges$fromJSONString(GroupChangesJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
GroupChanges$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GroupContents` *GroupContents*

Description

GroupContents Class

Format

An `R6Class` generator object

Public fields

`entries` list(GroupEntry) [optional]

`pagination_metadata` PaginationMetadata [optional]

Methods

Public methods:

- `GroupContents$new()`
- `GroupContents$toJSON()`
- `GroupContents$fromJSON()`
- `GroupContents$toJSONString()`
- `GroupContents$fromJSONString()`
- `GroupContents$clone()`

Method `new()`:

Usage:

```
GroupContents$new(entries = NULL, pagination_metadata = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
GroupContents$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
GroupContents$fromJSON(GroupContentsJson)
```

Method `toJSONString()`**:**

Usage:

```
GroupContents$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
GroupContents$fromJSONString(GroupContentsJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupContents$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GroupContentsFilterData` *GroupContentsFilterData*

Description

GroupContentsFilterData Class

Format

An `R6Class` generator object

Public fields

`namespaces` list(character) [optional]

Methods

Public methods:

- GroupContentsFilterData$new()
- GroupContentsFilterData$toJSON()
- GroupContentsFilterData$fromJSON()
- GroupContentsFilterData$toJSONString()
- GroupContentsFilterData$fromJSONString()
- GroupContentsFilterData$clone()

Method `new()`:

Usage:

```
GroupContentsFilterData$new(namespaces = NULL, ...)
```

Method `toJSON()`:

Usage:

```
GroupContentsFilterData$toJSON()
```

Method `fromJSON()`:

Usage:

```
GroupContentsFilterData$fromJSON(GroupContentsFilterDataJson)
```

Method `toJSONString()`:

Usage:

```
GroupContentsFilterData$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
GroupContentsFilterData$fromJSONString(GroupContentsFilterDataJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
GroupContentsFilterData$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

GroupCreate *GroupCreate*

Description

GroupCreate Class

Format

An R6Class generator object

Public fields

description character [optional]

name character [optional]

parent character [optional]

uri character [optional]

logo character [optional]

access_credentials_name character [optional]

tags list(character) [optional]

license_id character [optional]

license_text character [optional]

Methods

Public methods:

- GroupCreate$new()
- GroupCreate$toJSON()
- GroupCreate$fromJSON()
- GroupCreate$toJSONString()
- GroupCreate$fromJSONString()
- GroupCreate$clone()

Method new():

Usage:

```
GroupCreate$new(
  description = NULL,
  name = NULL,
  parent = NULL,
  uri = NULL,
  logo = NULL,
  access_credentials_name = NULL,
  tags = NULL,
```

```
  license_id = NULL,
  license_text = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
GroupCreate$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
GroupCreate$fromJSON(GroupCreateJson)
```

Method `toJSONString()`**:**

Usage:

```
GroupCreate$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
GroupCreate$fromJSONString(GroupCreateJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupCreate$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GroupEntry` *GroupEntry*

Description

GroupEntry Class

Format

An `R6Class` generator object

Public fields

`member_id` character [optional]

`group` GroupInfo [optional]

`array` ArrayInfo [optional]

Methods

Public methods:

- GroupEntry$new()
- GroupEntry$toJSON()
- GroupEntry$fromJSON()
- GroupEntry$toJSONString()
- GroupEntry$fromJSONString()
- GroupEntry$clone()

Method `new()`**:**

Usage:

```
GroupEntry$new(member_id = NULL, group = NULL, array = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
GroupEntry$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
GroupEntry$fromJSON(GroupEntryJson)
```

Method `toJSONString()`**:**

Usage:

```
GroupEntry$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
GroupEntry$fromJSONString(GroupEntryJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupEntry$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`GroupInfo` *GroupInfo*

Description

GroupInfo Class

Format

An `R6Class` generator object

Public fields

`id` character [optional]

`namespace` character [optional]

`name` character [optional]

`description` character [optional]

`uri` character [optional]

`tiledb_uri` character [optional]

`asset_count` numeric [optional]

`group_count` numeric [optional]

`size` numeric [optional]

`last_accessed` character [optional]

`allowed_actions` list(GroupActions) [optional]

`logo` character [optional]

`access_credentials_name` character [optional]

`share_count` numeric [optional]

`public_share` character [optional]

`tags` list(character) [optional]

`license_id` character [optional]

`license_text` character [optional]

Methods

Public methods:

- `GroupInfo$new()`
- `GroupInfo$toJSON()`
- `GroupInfo$fromJSON()`
- `GroupInfo$toJSONString()`
- `GroupInfo$fromJSONString()`
- `GroupInfo$clone()`

Method `new()`**:**

Usage:

```
GroupInfo$new(
  id = NULL,
  namespace = NULL,
  name = NULL,
  description = NULL,
  uri = NULL,
  tiledb_uri = NULL,
  asset_count = NULL,
  group_count = NULL,
  size = NULL,
  last_accessed = NULL,
  allowed_actions = NULL,
  logo = NULL,
  access_credentials_name = NULL,
  share_count = NULL,
  public_share = NULL,
  tags = NULL,
  license_id = NULL,
  license_text = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
GroupInfo$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
GroupInfo$fromJSON(GroupInfoJson)
```

Method `toJSONString()`**:**

Usage:

```
GroupInfo$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
GroupInfo$fromJSONString(GroupInfoJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupInfo$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

GroupListing *GroupListing*

Description

GroupListing Class

Format

An R6Class generator object

Public fields

`id` character [optional]

`namespace` character [optional]

`name` character [optional]

`description` character [optional]

`groups` list(Group) [optional]

`assets` list(ArrayInfo) [optional]

`pagination_metadata` PaginationMetadata [optional]

Methods

Public methods:

- `GroupListing$new()`
- `GroupListing$toJSON()`
- `GroupListing$fromJSON()`
- `GroupListing$toJSONString()`
- `GroupListing$fromJSONString()`
- `GroupListing$clone()`

Method `new()`:

Usage:

```
GroupListing$new(
  id = NULL,
  namespace = NULL,
  name = NULL,
  description = NULL,
  groups = NULL,
  assets = NULL,
  pagination_metadata = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
GroupListing$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
GroupListing$fromJSON(GroupListingJson)
```

Method `toJSONString()`**:**

Usage:

```
GroupListing$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
GroupListing$fromJSONString(GroupListingJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupListing$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GroupListingAllOf` *GroupListingAllOf*

Description

GroupListingAllOf Class

Format

An `R6Class` generator object

Public fields

`groups` list(Group) [optional]

`assets` list(ArrayInfo) [optional]

`pagination_metadata` PaginationMetadata [optional]

Methods

Public methods:

- GroupListingAllOf$new()
- GroupListingAllOf$toJSON()
- GroupListingAllOf$fromJSON()
- GroupListingAllOf$toJSONString()
- GroupListingAllOf$fromJSONString()
- GroupListingAllOf$clone()

Method `new()`:

Usage:

```
GroupListingAllOf$new(
  groups = NULL,
  assets = NULL,
  pagination_metadata = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
GroupListingAllOf$toJSON()
```

Method `fromJSON()`:

Usage:

```
GroupListingAllOf$fromJSON(GroupListingAllOfJson)
```

Method `toJSONString()`:

Usage:

```
GroupListingAllOf$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
GroupListingAllOf$fromJSONString(GroupListingAllOfJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
GroupListingAllOf$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

GroupMember *GroupMember*

Description

GroupMember Class

Format

An R6Class generator object

Public fields

`namespace` character [optional]

`name` character [optional]

`member_type` GroupMemberType [optional]

Methods

Public methods:

- `GroupMember$new()`
- `GroupMember$toJSON()`
- `GroupMember$fromJSON()`
- `GroupMember$toJSONString()`
- `GroupMember$fromJSONString()`
- `GroupMember$clone()`

Method `new()`:

Usage:

```
GroupMember$new(namespace = NULL, name = NULL, member_type = NULL, ...)
```

Method `toJSON()`:

Usage:

```
GroupMember$toJSON()
```

Method `fromJSON()`:

Usage:

```
GroupMember$fromJSON(GroupMemberJson)
```

Method `toJSONString()`:

Usage:

```
GroupMember$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
GroupMember$fromJSONString(GroupMemberJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupMember$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GroupMemberAssetType` *GroupMemberAssetType*

Description

GroupMemberAssetType Class

Format

An `R6Class` generator object

Methods

Public methods:

- `GroupMemberAssetType$new()`
- `GroupMemberAssetType$toJSON()`
- `GroupMemberAssetType$fromJSON()`
- `GroupMemberAssetType$toJSONString()`
- `GroupMemberAssetType$fromJSONString()`
- `GroupMemberAssetType$clone()`

Method `new()`**:**

Usage:

```
GroupMemberAssetType$new(...)
```

Method `toJSON()`**:**

Usage:

```
GroupMemberAssetType$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
GroupMemberAssetType$fromJSON(GroupMemberAssetTypeJson)
```

Method `toJSONString()`**:**

Usage:

```
GroupMemberAssetType$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
GroupMemberAssetType$fromJSONString(GroupMemberAssetTypeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupMemberAssetType$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GroupMemberType` *GroupMemberType*

Description

GroupMemberType Class

Format

An `R6Class` generator object

Methods

Public methods:

- `GroupMemberType$new()`
- `GroupMemberType$toJSON()`
- `GroupMemberType$fromJSON()`
- `GroupMemberType$toJSONString()`
- `GroupMemberType$fromJSONString()`
- `GroupMemberType$clone()`

Method `new()`**:**

Usage:

```
GroupMemberType$new(...)
```

Method `toJSON()`**:**

Usage:

```
GroupMemberType$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
GroupMemberType$fromJSON(GroupMemberTypeJson)
```

Method `toJSONString()`**:**

Usage:

```
GroupMemberType$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
GroupMemberType$fromJSONString(GroupMemberTypeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupMemberType$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GroupRegister` *GroupRegister*

Description

GroupRegister Class

Format

An `R6Class` generator object

Public fields

`description` character [optional]

`name` character [optional]

`parent` character [optional]

`uri` character [optional]

`logo` character [optional]

`access_credentials_name` character [optional]

`tags` list(character) [optional]

`license_id` character [optional]

`license_text` character [optional]

Methods

Public methods:

- GroupRegister$new()
- GroupRegister$toJSON()
- GroupRegister$fromJSON()
- GroupRegister$toJSONString()
- GroupRegister$fromJSONString()
- GroupRegister$clone()

Method `new()`**:**

Usage:

```
GroupRegister$new(
  description = NULL,
  name = NULL,
  parent = NULL,
  uri = NULL,
  logo = NULL,
  access_credentials_name = NULL,
  tags = NULL,
  license_id = NULL,
  license_text = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
GroupRegister$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
GroupRegister$fromJSON(GroupRegisterJson)
```

Method `toJSONString()`**:**

Usage:

```
GroupRegister$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
GroupRegister$fromJSONString(GroupRegisterJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupRegister$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GroupsApi` *Groups operations*

Description

tiledbcloud.Groups

Format

An `R6Class` generator object

Methods

ChangeGroupContents Changes the contents of the group by adding/removing members.

@*param* group.namespace character

- @*param* group.name character
- @*param* group.changes GroupChanges
- status code : 204 | all changes applied successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

CreateGroup Creates a new group in the namespace.

@*param* namespace character

- @*param* group.create GroupCreate
- status code : 204 | group created successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

DeleteGroup Deletes the group. The assets are not deleted nor are not relocated to any other group

@*param* group.namespace character

- @*param* group.name character
- status code : 204 | group deleted successfully

- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetGroup Returns the the group

@*param* group.namespace character

- @*param* group.name character
- @*returnType* GroupInfo

- status code : 200 | the group metadata
- return type : GroupInfo
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetGroupContents Returns the contents of the group

@*param* group.namespace character

- @*param* group.name character
- @*param* page integer
- @*param* per.page integer
- @*param* namespace character
- @*param* search character
- @*param* orderby character
- @*param* tag list(character)
- @*param* exclude.tag list(character)
- @*param* member.type list(character)
- @*param* exclude.member.type list(character)
- @*returnType* GroupContents

- status code : 200 | the group contents
- return type : GroupContents
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

GetGroupSharingPolicies Get all sharing details of the group

@*param* group.namespace character

- @*param* group.name character
- @*returnType* list(GroupSharing)

- status code : 200 | List of all specific sharing policies
- return type : array[GroupSharing]
- response headers :

- status code : 404 | Group does not exist or user does not have permissions to view group-sharing policies
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GroupsBrowserOwnedFiltersGet Fetch data to initialize filters for the groups browser

@*returnType* GroupBrowserFilterData

- status code : 200 | Filter data
- return type : GroupBrowserFilterData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GroupsBrowserPublicFiltersGet Fetch data to initialize filters for the groups browser

@*returnType* GroupBrowserFilterData

- status code : 200 | Filter data
- return type : GroupBrowserFilterData
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

GroupsBrowserSharedFiltersGet Fetch data to initialize filters for the groups browser

@*returnType* GroupBrowserFilterData

- status code : 200 | Filter data
- return type : GroupBrowserFilterData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GroupsGroupNamespaceGroupNameContentsFiltersGet Fetch data to initialize filters for the group contents

@*param* group.namespace character

- @*param* group.name character
- @*returnType* GroupContentsFilterData

- status code : 200 | Filter data
- return type : GroupContentsFilterData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ListOwnedGroups Returns one page of owned groups.

@*param* page integer

- @*param* per.page integer
- @*param* search character
- @*param* namespace character
- @*param* orderby character
- @*param* permissions character
- @*param* tag list(character)
- @*param* exclude.tag list(character)
- @*param* flat character
- @*param* parent character
- @*returnType* GroupBrowserData

- status code : 200 | the group contents
- return type : GroupBrowserData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ListPublicGroups Returns one page of public groups.

@*param* page integer
- @*param* per.page integer
- @*param* search character
- @*param* namespace character
- @*param* orderby character
- @*param* permissions character
- @*param* tag list(character)
- @*param* exclude.tag list(character)
- @*param* flat character
- @*param* parent character
- @*returnType* GroupBrowserData

- status code : 200 | the group contents
- return type : GroupBrowserData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ListSharedGroups Returns one page of shared groups.

@*param* page integer
- @*param* per.page integer
- @*param* search character
- @*param* namespace character
- @*param* orderby character
- @*param* permissions character
- @*param* tag list(character)
- @*param* exclude.tag list(character)
- @*param* flat character
- @*param* parent character

- @*param* shared.to list(character)
- @*returnType* GroupBrowserData

- status code : 200 | the group contents
- return type : GroupBrowserData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

RegisterGroup Registers an existing group in the namespace.

@*param* namespace character

- @*param* array character
- @*param* group.register GroupRegister
- status code : 204 | group created successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ShareGroup Share a group with a namespace

@*param* group.namespace character

- @*param* group.name character
- @*param* group.sharing.request GroupSharingRequest
- status code : 204 | Group shared successfully
- response headers :

- status code : 404 | Group does not exist or user does not have permissions to share group
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

UpdateGroup Changes attributes of the group

@*param* group.namespace character

- @*param* group.name character

- *@param* group.update GroupUpdate
- status code : 204 | attributes changed successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `GroupsApi$new()`
- `GroupsApi$ChangeGroupContents()`
- `GroupsApi$ChangeGroupContentsWithHttpInfo()`
- `GroupsApi$CreateGroup()`
- `GroupsApi$CreateGroupWithHttpInfo()`
- `GroupsApi$DeleteGroup()`
- `GroupsApi$DeleteGroupWithHttpInfo()`
- `GroupsApi$GetGroup()`
- `GroupsApi$GetGroupWithHttpInfo()`
- `GroupsApi$GetGroupContents()`
- `GroupsApi$GetGroupContentsWithHttpInfo()`
- `GroupsApi$GetGroupSharingPolicies()`
- `GroupsApi$GetGroupSharingPoliciesWithHttpInfo()`
- `GroupsApi$GroupsBrowserOwnedFiltersGet()`
- `GroupsApi$GroupsBrowserOwnedFiltersGetWithHttpInfo()`
- `GroupsApi$GroupsBrowserPublicFiltersGet()`
- `GroupsApi$GroupsBrowserPublicFiltersGetWithHttpInfo()`
- `GroupsApi$GroupsBrowserSharedFiltersGet()`
- `GroupsApi$GroupsBrowserSharedFiltersGetWithHttpInfo()`
- `GroupsApi$GroupsGroupNamespaceGroupNameContentsFiltersGet()`
- `GroupsApi$GroupsGroupNamespaceGroupNameContentsFiltersGetWithHttpInfo()`
- `GroupsApi$ListOwnedGroups()`
- `GroupsApi$ListOwnedGroupsWithHttpInfo()`
- `GroupsApi$ListPublicGroups()`
- `GroupsApi$ListPublicGroupsWithHttpInfo()`
- `GroupsApi$ListSharedGroups()`
- `GroupsApi$ListSharedGroupsWithHttpInfo()`

- GroupsApi$RegisterGroup()
- GroupsApi$RegisterGroupWithHttpInfo()
- GroupsApi$ShareGroup()
- GroupsApi$ShareGroupWithHttpInfo()
- GroupsApi$UpdateGroup()
- GroupsApi$UpdateGroupWithHttpInfo()
- GroupsApi$clone()

Method `new()`**:**

Usage:

```
GroupsApi$new(apiClient)
```

Method `ChangeGroupContents()`**:**

Usage:

```
GroupsApi$ChangeGroupContents(
  group.namespace,
  group.name,
  group.changes = NULL,
  ...
)
```

Method `ChangeGroupContentsWithHttpInfo()`**:**

Usage:

```
GroupsApi$ChangeGroupContentsWithHttpInfo(
  group.namespace,
  group.name,
  group.changes = NULL,
  ...
)
```

Method `CreateGroup()`**:**

Usage:

```
GroupsApi$CreateGroup(namespace, group.create = NULL, ...)
```

Method `CreateGroupWithHttpInfo()`**:**

Usage:

```
GroupsApi$CreateGroupWithHttpInfo(namespace, group.create = NULL, ...)
```

Method `DeleteGroup()`**:**

Usage:

```
GroupsApi$DeleteGroup(group.namespace, group.name, ...)
```

Method `DeleteGroupWithHttpInfo()`**:**

Usage:

```
GroupsApi$DeleteGroupWithHttpInfo(group.namespace, group.name, ...)
```

Method `GetGroup()`**:**

Usage:

```
GroupsApi$GetGroup(group.namespace, group.name, ...)
```

Method `GetGroupWithHttpInfo()`**:**

Usage:

```
GroupsApi$GetGroupWithHttpInfo(group.namespace, group.name, ...)
```

Method `GetGroupContents()`**:**

Usage:

```
GroupsApi$GetGroupContents(
  group.namespace,
  group.name,
  page = NULL,
  per.page = NULL,
  namespace = NULL,
  search = NULL,
  orderby = NULL,
  tag = NULL,
  exclude.tag = NULL,
  member.type = NULL,
  exclude.member.type = NULL,
  ...
)
```

Method `GetGroupContentsWithHttpInfo()`**:**

Usage:

```
GroupsApi$GetGroupContentsWithHttpInfo(
  group.namespace,
  group.name,
  page = NULL,
  per.page = NULL,
  namespace = NULL,
  search = NULL,
  orderby = NULL,
  tag = NULL,
  exclude.tag = NULL,
  member.type = NULL,
  exclude.member.type = NULL,
  ...
)
```

Method `GetGroupSharingPolicies()`**:**

Usage:

```
GroupsApi$GetGroupSharingPolicies(group.namespace, group.name, ...)
```

Method `GetGroupSharingPoliciesWithHttpInfo()`**:**

Usage:

```
GroupsApi$GetGroupSharingPoliciesWithHttpInfo(group.namespace, group.name, ...)
```

Method `GroupsBrowserOwnedFiltersGet()`**:**

Usage:

```
GroupsApi$GroupsBrowserOwnedFiltersGet(...)
```

Method `GroupsBrowserOwnedFiltersGetWithHttpInfo()`**:**

Usage:

```
GroupsApi$GroupsBrowserOwnedFiltersGetWithHttpInfo(...)
```

Method `GroupsBrowserPublicFiltersGet()`**:**

Usage:

```
GroupsApi$GroupsBrowserPublicFiltersGet(...)
```

Method `GroupsBrowserPublicFiltersGetWithHttpInfo()`**:**

Usage:

```
GroupsApi$GroupsBrowserPublicFiltersGetWithHttpInfo(...)
```

Method `GroupsBrowserSharedFiltersGet()`**:**

Usage:

```
GroupsApi$GroupsBrowserSharedFiltersGet(...)
```

Method `GroupsBrowserSharedFiltersGetWithHttpInfo()`**:**

Usage:

```
GroupsApi$GroupsBrowserSharedFiltersGetWithHttpInfo(...)
```

Method `GroupsGroupNamespaceGroupNameContentsFiltersGet()`**:**

Usage:

```
GroupsApi$GroupsGroupNamespaceGroupNameContentsFiltersGet(
  group.namespace,
  group.name,
  ...
)
```

Method `GroupsGroupNamespaceGroupNameContentsFiltersGetWithHttpInfo()`**:**

Usage:

```
GroupsApi$GroupsGroupNamespaceGroupNameContentsFiltersGetWithHttpInfo(
  group.namespace,
  group.name,
  ...
)
```

Method `ListOwnedGroups()`**:**

Usage:

```
GroupsApi$ListOwnedGroups(
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  flat = NULL,
  parent = NULL,
  ...
)
```

Method `ListOwnedGroupsWithHttpInfo()`**:**

Usage:

```
GroupsApi$ListOwnedGroupsWithHttpInfo(
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  flat = NULL,
  parent = NULL,
  ...
)
```

Method `ListPublicGroups()`**:**

Usage:

```
GroupsApi$ListPublicGroups(
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  flat = NULL,
  parent = NULL,
  ...
)
```

Method `ListPublicGroupsWithHttpInfo()`**:**

Usage:

```
GroupsApi$ListPublicGroupsWithHttpInfo(
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  flat = NULL,
  parent = NULL,
  ...
)
```

Method `ListSharedGroups()`**:**

Usage:

```
GroupsApi$ListSharedGroups(
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  flat = NULL,
  parent = NULL,
  shared.to = NULL,
  ...
)
```

Method `ListSharedGroupsWithHttpInfo()`**:**

Usage:

```
GroupsApi$ListSharedGroupsWithHttpInfo(
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  flat = NULL,
  parent = NULL,
  shared.to = NULL,
  ...
)
```

Method `RegisterGroup()`**:**

Usage:

```
GroupsApi$RegisterGroup(namespace, array, group.register = NULL, ...)
```

Method `RegisterGroupWithHttpInfo()`**:**

Usage:

```
GroupsApi$RegisterGroupWithHttpInfo(
  namespace,
  array,
  group.register = NULL,
  ...
)
```

Method `ShareGroup()`**:**

Usage:

```
GroupsApi$ShareGroup(group.namespace, group.name, group.sharing.request, ...)
```

Method `ShareGroupWithHttpInfo()`**:**

Usage:

```
GroupsApi$ShareGroupWithHttpInfo(
  group.namespace,
  group.name,
  group.sharing.request,
  ...
)
```

Method `UpdateGroup()`**:**

Usage:

```
GroupsApi$UpdateGroup(group.namespace, group.name, group.update = NULL, ...)
```

Method `UpdateGroupWithHttpInfo()`**:**

Usage:

```
GroupsApi$UpdateGroupWithHttpInfo(
  group.namespace,
  group.name,
  group.update = NULL,
  ...
)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupsApi$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
####################  ChangeGroupContents  ####################

library(tiledbcloud)
var.group.namespace <- 'group.namespace_example' # character | The namespace of the group
var.group.name <- 'group.name_example' # character | The unique name or id of the group
var.group.changes <- GroupChanges$new() # GroupChanges |

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ChangeGroupContents(var.group.namespace, var.group.name, group.changes=var.group.changes)


####################  CreateGroup  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the group
var.group.create <- GroupCreate$new() # GroupCreate |

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$CreateGroup(var.namespace, group.create=var.group.create)


####################  DeleteGroup  ####################

library(tiledbcloud)
var.group.namespace <- 'group.namespace_example' # character | The namespace of the group
var.group.name <- 'group.name_example' # character | The unique name or id of the group

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
```

```
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteGroup(var.group.namespace, var.group.name)


####################  GetGroup  ####################

library(tiledbcloud)
var.group.namespace <- 'group.namespace_example' # character | The namespace of the group
var.group.name <- 'group.name_example' # character | The unique name or id of the group

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetGroup(var.group.namespace, var.group.name)


####################  GetGroupContents  ####################

library(tiledbcloud)
var.group.namespace <- 'group.namespace_example' # character | The namespace of the group
var.group.name <- 'group.name_example' # character | The unique name or id of the group
var.page <- 56 # integer | pagination offset for assets
var.per.page <- 56 # integer | pagination limit for assets
var.namespace <- 'namespace_example' # character | namespace to search for
var.search <- 'search_example' # character | search string that will look at name, namespace or description fields
var.orderby <- 'orderby_example' # character | sort by which field valid values include last_accessed, size, name
var.tag <- ['tag_example'] # array[character] | tag to search for, more than one can be included
var.exclude.tag <- ['exclude.tag_example'] # array[character] | tags to exclude matching array in results, more tha
var.member.type <- ['member.type_example'] # array[character] | member type to search for, more than one can be incl
var.exclude.member.type <- ['exclude.member.type_example'] # array[character] | member type to exclude matching gr

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
```

```
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetGroupContents(var.group.namespace, var.group.name, page=var.page, per.page=var.per.pag


####################  GetGroupSharingPolicies  ####################

library(tiledbcloud)
var.group.namespace <- 'group.namespace_example' # character | The namespace of the group
var.group.name <- 'group.name_example' # character | The unique name or id of the group

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetGroupSharingPolicies(var.group.namespace, var.group.name)


####################  GroupsBrowserOwnedFiltersGet  ####################

library(tiledbcloud)

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GroupsBrowserOwnedFiltersGet()


####################  GroupsBrowserPublicFiltersGet  ####################

library(tiledbcloud)

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';
```

```
#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GroupsBrowserPublicFiltersGet()


####################  GroupsBrowserSharedFiltersGet  ####################

library(tiledbcloud)

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GroupsBrowserSharedFiltersGet()


#################### GroupsGroupNamespaceGroupNameContentsFiltersGet ####################

library(tiledbcloud)
var.group.namespace <- 'group.namespace_example' # character | The namespace of the group
var.group.name <- 'group.name_example' # character | The unique name or id of the group

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GroupsGroupNamespaceGroupNameContentsFiltersGet(var.group.namespace, var.group.name)


####################  ListOwnedGroups  ####################

library(tiledbcloud)
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit
```

```
var.search <- 'search_example' # character | search string that will look at name, namespace or description fields
var.namespace <- 'namespace_example' # character | namespace
var.orderby <- 'orderby_example' # character | sort by which field valid values include last_accessed, size, name
var.permissions <- 'permissions_example' # character | permissions valid values include read, read_write, write, ac
var.tag <- ['tag_example'] # array[character] | tag to search for, more than one can be included
var.exclude.tag <- ['exclude.tag_example'] # array[character] | tags to exclude matching array in results, more tha
var.flat <- 'flat_example' # character | if true, ignores the nesting of groups and searches all of them
var.parent <- 'parent_example' # character | search only the children of the groups with this uuid

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ListOwnedGroups(page=var.page, per.page=var.per.page, search=var.search, namespace=var.na


####################  ListPublicGroups  ####################

library(tiledbcloud)
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit
var.search <- 'search_example' # character | search string that will look at name, namespace or description fields
var.namespace <- 'namespace_example' # character | namespace
var.orderby <- 'orderby_example' # character | sort by which field valid values include last_accessed, size, name
var.permissions <- 'permissions_example' # character | permissions valid values include read, read_write, write, ac
var.tag <- ['tag_example'] # array[character] | tag to search for, more than one can be included
var.exclude.tag <- ['exclude.tag_example'] # array[character] | tags to exclude matching array in results, more tha
var.flat <- 'flat_example' # character | if true, ignores the nesting of groups and searches all of them
var.parent <- 'parent_example' # character | search only the children of the groups with this uuid

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ListPublicGroups(page=var.page, per.page=var.per.page, search=var.search, namespace=var.n


####################  ListSharedGroups  ####################
```

```
library(tiledbcloud)
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit
var.search <- 'search_example' # character | search string that will look at name, namespace or description fields
var.namespace <- 'namespace_example' # character | namespace
var.orderby <- 'orderby_example' # character | sort by which field valid values include last_accessed, size, name
var.permissions <- 'permissions_example' # character | permissions valid values include read, read_write, write, ad
var.tag <- ['tag_example'] # array[character] | tag to search for, more than one can be included
var.exclude.tag <- ['exclude.tag_example'] # array[character] | tags to exclude matching array in results, more tha
var.flat <- 'flat_example' # character | if true, ignores the nesting of groups and searches all of them
var.parent <- 'parent_example' # character | search only the children of the groups with this uuid
var.shared.to <- ['shared.to_example'] # array[character] | namespaces to filter results of where there groups were

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ListSharedGroups(page=var.page, per.page=var.per.page, search=var.search, namespace=var.n


####################  RegisterGroup  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace of the group
var.array <- 'array_example' # character | The unique name or id of the group
var.group.register <- GroupRegister$new() # GroupRegister |

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$RegisterGroup(var.namespace, var.array, group.register=var.group.register)


####################  ShareGroup  ####################

library(tiledbcloud)
var.group.namespace <- 'group.namespace_example' # character | The namespace of the group
var.group.name <- 'group.name_example' # character | The unique name or id of the group
```

```
var.group.sharing.request <- GroupSharingRequest$new() # GroupSharingRequest | Namespace and list of permissions t

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ShareGroup(var.group.namespace, var.group.name, var.group.sharing.request)


####################  UpdateGroup  ####################

library(tiledbcloud)
var.group.namespace <- 'group.namespace_example' # character | The namespace of the group
var.group.name <- 'group.name_example' # character | The unique name or id of the group
var.group.update <- GroupUpdate$new() # GroupUpdate |

api.instance <- GroupsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$UpdateGroup(var.group.namespace, var.group.name, group.update=var.group.update)



## End(Not run)
```

GroupSharing *GroupSharing*

Description

GroupSharing Class

Format

An R6Class generator object

Public fields

`group_actions` list(GroupActions) [optional]

`array_actions` list(ArrayActions) [optional]

`namespace` character [optional]

`namespace_type` character [optional]

Methods

Public methods:

- `GroupSharing$new()`
- `GroupSharing$toJSON()`
- `GroupSharing$fromJSON()`
- `GroupSharing$toJSONString()`
- `GroupSharing$fromJSONString()`
- `GroupSharing$clone()`

Method `new()`:

Usage:

```
GroupSharing$new(
  group_actions = NULL,
  array_actions = NULL,
  namespace = NULL,
  namespace_type = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
GroupSharing$toJSON()
```

Method `fromJSON()`:

Usage:

```
GroupSharing$fromJSON(GroupSharingJson)
```

Method `toJSONString()`:

Usage:

```
GroupSharing$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
GroupSharing$fromJSONString(GroupSharingJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
GroupSharing$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

GroupSharingRequest *GroupSharingRequest*

Description

GroupSharingRequest Class

Format

An R6Class generator object

Public fields

group_actions list(GroupActions) [optional]

array_actions list(ArrayActions) [optional]

namespace character [optional]

Methods

Public methods:

- GroupSharingRequest$new()
- GroupSharingRequest$toJSON()
- GroupSharingRequest$fromJSON()
- GroupSharingRequest$toJSONString()
- GroupSharingRequest$fromJSONString()
- GroupSharingRequest$clone()

Method new():

Usage:

```
GroupSharingRequest$new(
  group_actions = NULL,
  array_actions = NULL,
  namespace = NULL,
  ...
)
```

Method toJSON():

Usage:

```
GroupSharingRequest$toJSON()
```

Method fromJSON():

Usage:

```
GroupSharingRequest$fromJSON(GroupSharingRequestJson)
```

Method toJSONString():

Usage:

```
GroupSharingRequest$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
GroupSharingRequest$fromJSONString(GroupSharingRequestJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupSharingRequest$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GroupUpdate` *GroupUpdate*

Description

GroupUpdate Class

Format

An `R6Class` generator object

Public fields

`description` character [optional]

`name` character [optional]

`logo` character [optional]

`access_credentials_name` character [optional]

`tags` list(character) [optional]

`license_id` character [optional]

`license_text` character [optional]

Methods

Public methods:

- `GroupUpdate$new()`
- `GroupUpdate$toJSON()`
- `GroupUpdate$fromJSON()`
- `GroupUpdate$toJSONString()`
- `GroupUpdate$fromJSONString()`
- `GroupUpdate$clone()`

Method `new()`**:**

Usage:

```
GroupUpdate$new(
  description = NULL,
  name = NULL,
  logo = NULL,
  access_credentials_name = NULL,
  tags = NULL,
  license_id = NULL,
  license_text = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
GroupUpdate$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
GroupUpdate$fromJSON(GroupUpdateJson)
```

Method `toJSONString()`**:**

Usage:

```
GroupUpdate$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
GroupUpdate$fromJSONString(GroupUpdateJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GroupUpdate$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`group_info` *Group Information*

Description

Show information about a group on TileDB Cloud.

Usage

```
group_info(namespace, name)
```

Arguments

namespace	Like "TileDB-Inc"
name	Name of the group

Value

A list of properties

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_udf(), execute_sql_query(), get_udf_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), update_udf_info(), user_profile()

InlineObject *InlineObject*

Description

InlineObject Class

Format

An R6Class generator object

Public fields

password character [optional]

Methods

Public methods:

- InlineObject$new()
- InlineObject$toJSON()
- InlineObject$fromJSON()
- InlineObject$toJSONString()
- InlineObject$fromJSONString()
- InlineObject$clone()

Method new():

Usage:

```
InlineObject$new(password = NULL, ...)
```

Method toJSON():

Usage:

```
InlineObject$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
InlineObject$fromJSON(InlineObjectJson)
```

Method `toJSONString()`**:**

Usage:

```
InlineObject$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
InlineObject$fromJSONString(InlineObjectJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
InlineObject$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`InlineResponse200` *InlineResponse200*

Description

InlineResponse200 Class

Format

An `R6Class` generator object

Public fields

`stats` character [optional]

Methods

Public methods:

- `InlineResponse200$new()`
- `InlineResponse200$toJSON()`
- `InlineResponse200$fromJSON()`
- `InlineResponse200$toJSONString()`
- `InlineResponse200$fromJSONString()`
- `InlineResponse200$clone()`

Method `new()`**:**

Usage:

```
InlineResponse200$new(stats = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
InlineResponse200$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
InlineResponse200$fromJSON(InlineResponse200Json)
```

Method `toJSONString()`**:**

Usage:

```
InlineResponse200$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
InlineResponse200$fromJSONString(InlineResponse200Json)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
InlineResponse200$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Invitation` *Invitation*

Description

Invitation Class

Format

An `R6Class` generator object

Public fields

`id` character [optional]

`invitation_type` InvitationType [optional]

`owner_namespace_uuid` character [optional]

`user_namespace_uuid` character [optional]

`organization_user_uuid` character [optional]

`organization_name` character [optional]

`organization_role` OrganizationRoles [optional]

`array_uuid` character [optional]

`array_name` character [optional]

`email` character [optional]

`actions` character [optional]

`status` InvitationStatus [optional]

`created_at` character [optional]

`expires_at` character [optional]

`accepted_at` character [optional]

Methods

Public methods:

- `Invitation$new()`
- `Invitation$toJSON()`
- `Invitation$fromJSON()`
- `Invitation$toJSONString()`
- `Invitation$fromJSONString()`
- `Invitation$clone()`

Method `new()`:

Usage:

```
Invitation$new(
  id = NULL,
  invitation_type = NULL,
  owner_namespace_uuid = NULL,
  user_namespace_uuid = NULL,
  organization_user_uuid = NULL,
  organization_name = NULL,
  organization_role = NULL,
  array_uuid = NULL,
  array_name = NULL,
  email = NULL,
  actions = NULL,
  status = NULL,
```

```
  created_at = NULL,
  expires_at = NULL,
  accepted_at = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
Invitation$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
Invitation$fromJSON(InvitationJson)
```

Method `toJSONString()`**:**

Usage:

```
Invitation$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Invitation$fromJSONString(InvitationJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Invitation$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`InvitationApi` *Invitation operations*

Description

tiledbcloud.Invitation

Format

An `R6Class` generator object

Methods

AcceptInvitation Accepts invitation

@*param* invitation character

- status code : 204 | Invitation was accepted successfully
- response headers :

- status code : 404 | Could not find invitation identifier
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

CancelJoinOrganization Cancels join organization invitation

@*param* invitation character

- @*param* organization character
- status code : 204 | Invitation cancelled successfully
- response headers :

- status code : 404 | No invitation was found to cancel
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

CancelShareArrayByInvite Cancels array sharing invitation

@*param* namespace character

- @*param* invitation character
- @*param* array character
- status code : 204 | Invitation cancelled successfully
- response headers :

- status code : 404 | No invitation was found to cancel
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

FetchInvitations Fetch a list of invitations

@*param* organization character

- @*param* array character
- @*param* start integer
- @*param* end integer
- @*param* page integer
- @*param* per.page integer
- @*param* type character
- @*param* status character
- @*param* orderby character
- @*returnType* InvitationData

- status code : 200 | List of invitations and pagination metadata
- return type : InvitationData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

JoinOrganization Sends email to multiple recipients with joining information regarding an organization

@*param* organization character

- @*param* email.invite InvitationOrganizationJoinEmail
- status code : 204 | Email sent successfully to user for email confirmation link
- response headers :

- status code : 404 | Could not reach one or more recipients
- return type : InvitationOrganizationJoinEmail
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ShareArrayByInvite Sends email to multiple recipients with sharing information regarding an array

@*param* namespace character

- @*param* array character
- @*param* email.invite InvitationArrayShareEmail
- status code : 204 | Email sent successfully to user for email confirmation link
- response headers :

- status code : 404 | Could not reach one or more recipients
- return type : InvitationArrayShareEmail
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `InvitationApi$new()`
- `InvitationApi$AcceptInvitation()`
- `InvitationApi$AcceptInvitationWithHttpInfo()`
- `InvitationApi$CancelJoinOrganization()`
- `InvitationApi$CancelJoinOrganizationWithHttpInfo()`
- `InvitationApi$CancelShareArrayByInvite()`
- `InvitationApi$CancelShareArrayByInviteWithHttpInfo()`
- `InvitationApi$FetchInvitations()`
- `InvitationApi$FetchInvitationsWithHttpInfo()`
- `InvitationApi$JoinOrganization()`
- `InvitationApi$JoinOrganizationWithHttpInfo()`
- `InvitationApi$ShareArrayByInvite()`
- `InvitationApi$ShareArrayByInviteWithHttpInfo()`
- `InvitationApi$clone()`

Method `new()`:

Usage:

```
InvitationApi$new(apiClient)
```

Method `AcceptInvitation()`:

Usage:

```
InvitationApi$AcceptInvitation(invitation, ...)
```

Method `AcceptInvitationWithHttpInfo()`**:**

Usage:

```
InvitationApi$AcceptInvitationWithHttpInfo(invitation, ...)
```

Method `CancelJoinOrganization()`**:**

Usage:

```
InvitationApi$CancelJoinOrganization(invitation, organization, ...)
```

Method `CancelJoinOrganizationWithHttpInfo()`**:**

Usage:

```
InvitationApi$CancelJoinOrganizationWithHttpInfo(invitation, organization, ...)
```

Method `CancelShareArrayByInvite()`**:**

Usage:

```
InvitationApi$CancelShareArrayByInvite(namespace, invitation, array, ...)
```

Method `CancelShareArrayByInviteWithHttpInfo()`**:**

Usage:

```
InvitationApi$CancelShareArrayByInviteWithHttpInfo(
  namespace,
  invitation,
  array,
  ...
)
```

Method `FetchInvitations()`**:**

Usage:

```
InvitationApi$FetchInvitations(
  organization = NULL,
  array = NULL,
  start = NULL,
  end = NULL,
  page = NULL,
  per.page = NULL,
  type = NULL,
  status = NULL,
  orderby = NULL,
  ...
)
```

Method `FetchInvitationsWithHttpInfo()`**:**

Usage:

```
InvitationApi$FetchInvitationsWithHttpInfo(
  organization = NULL,
  array = NULL,
  start = NULL,
```

```
  end = NULL,
  page = NULL,
  per.page = NULL,
  type = NULL,
  status = NULL,
  orderby = NULL,
  ...
)
```

Method `JoinOrganization()`**:**

Usage:

```
InvitationApi$JoinOrganization(organization, email.invite, ...)
```

Method `JoinOrganizationWithHttpInfo()`**:**

Usage:

```
InvitationApi$JoinOrganizationWithHttpInfo(organization, email.invite, ...)
```

Method `ShareArrayByInvite()`**:**

Usage:

```
InvitationApi$ShareArrayByInvite(namespace, array, email.invite, ...)
```

Method `ShareArrayByInviteWithHttpInfo()`**:**

Usage:

```
InvitationApi$ShareArrayByInviteWithHttpInfo(
  namespace,
  array,
  email.invite,
  ...
)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
InvitationApi$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
####################  AcceptInvitation  ####################

library(tiledbcloud)
var.invitation <- 'invitation_example' # character | the ID of invitation about to be accepted

api.instance <- InvitationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';
```

```
#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$AcceptInvitation(var.invitation)


####################  CancelJoinOrganization  ####################

library(tiledbcloud)
var.invitation <- 'invitation_example' # character | the ID of invitation about to be cancelled
var.organization <- 'organization_example' # character | name or UUID of organization

api.instance <- InvitationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$CancelJoinOrganization(var.invitation, var.organization)


####################  CancelShareArrayByInvite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.invitation <- 'invitation_example' # character | the ID of invitation about to be cancelled
var.array <- 'array_example' # character | name/uri of array that is url-encoded

api.instance <- InvitationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$CancelShareArrayByInvite(var.namespace, var.invitation, var.array)


####################  FetchInvitations  ####################
```

```
library(tiledbcloud)
var.organization <- 'organization_example' # character | name or ID of organization to filter
var.array <- 'array_example' # character | name/uri of array that is url-encoded to filter
var.start <- 56 # integer | start time for tasks to filter by
var.end <- 56 # integer | end time for tasks to filter by
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit
var.type <- 'type_example' # character | invitation type, \"ARRAY_SHARE\", \"JOIN_ORGANIZATION\"
var.status <- 'status_example' # character | Filter to only return \"PENDING\", \"ACCEPTED\"
var.orderby <- 'orderby_example' # character | sort by which field valid values include timestamp, array_name, orga

api.instance <- InvitationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$FetchInvitations(organization=var.organization, array=var.array, start=var.start, end=var


####################  JoinOrganization  ####################

library(tiledbcloud)
var.organization <- 'organization_example' # character | name or UUID of organization
var.email.invite <- InvitationOrganizationJoinEmail$new() # InvitationOrganizationJoinEmail | list of email recip

api.instance <- InvitationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$JoinOrganization(var.organization, var.email.invite)


####################  ShareArrayByInvite  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.email.invite <- InvitationArrayShareEmail$new() # InvitationArrayShareEmail | list of email recipients

api.instance <- InvitationApi$new()
```

```
#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ShareArrayByInvite(var.namespace, var.array, var.email.invite)



## End(Not run)
```

`InvitationArrayShareEmail` *InvitationArrayShareEmail*

Description

InvitationArrayShareEmail Class

Format

An `R6Class` generator object

Public fields

`actions` list(ArrayActions)

`invitee_email` list(character)

Methods

Public methods:

- `InvitationArrayShareEmail$new()`
- `InvitationArrayShareEmail$toJSON()`
- `InvitationArrayShareEmail$fromJSON()`
- `InvitationArrayShareEmail$toJSONString()`
- `InvitationArrayShareEmail$fromJSONString()`
- `InvitationArrayShareEmail$clone()`

Method `new()`:

Usage:

```
InvitationArrayShareEmail$new(actions, invitee_email, ...)
```

Method `toJSON()`:

Usage:

```
InvitationArrayShareEmail$toJSON()
```

Method `fromJSON()`:

Usage:

```
InvitationArrayShareEmail$fromJSON(InvitationArrayShareEmailJson)
```

Method `toJSONString()`:

Usage:

```
InvitationArrayShareEmail$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
InvitationArrayShareEmail$fromJSONString(InvitationArrayShareEmailJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
InvitationArrayShareEmail$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`InvitationData` *InvitationData*

Description

InvitationData Class

Format

An `R6Class` generator object

Public fields

`invitations` list(Invitation) [optional]

`pagination_metadata` PaginationMetadata [optional]

Methods

Public methods:

Method `new()`:

Usage:

```
InvitationData$new(invitations = NULL, pagination_metadata = NULL, ...)
```

Method `toJSON()`:

Usage:

```
InvitationData$toJSON()
```

Method `fromJSON()`:

Usage:

```
InvitationData$fromJSON(InvitationDataJson)
```

Method `toJSONString()`:

Usage:

```
InvitationData$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
InvitationData$fromJSONString(InvitationDataJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
InvitationData$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

InvitationOrganizationJoinEmail
InvitationOrganizationJoinEmail

Description

InvitationOrganizationJoinEmail Class

Format

An R6Class generator object

Public fields

`actions` list(NamespaceActions) [optional]

`organization_role` OrganizationRoles

`invitee_email` list(character)

Methods

Public methods:

- `InvitationOrganizationJoinEmail$new()`
- `InvitationOrganizationJoinEmail$toJSON()`
- `InvitationOrganizationJoinEmail$fromJSON()`
- `InvitationOrganizationJoinEmail$toJSONString()`
- `InvitationOrganizationJoinEmail$fromJSONString()`
- `InvitationOrganizationJoinEmail$clone()`

Method `new()`:

Usage:

```
InvitationOrganizationJoinEmail$new(
  organization_role,
  invitee_email,
  actions = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
InvitationOrganizationJoinEmail$toJSON()
```

Method `fromJSON()`:

Usage:

```
InvitationOrganizationJoinEmail$fromJSON(InvitationOrganizationJoinEmailJson)
```

Method `toJSONString()`**:**

Usage:

```
InvitationOrganizationJoinEmail$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
InvitationOrganizationJoinEmail$fromJSONString(
  InvitationOrganizationJoinEmailJson
)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
InvitationOrganizationJoinEmail$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`InvitationStatus` *InvitationStatus*

Description

InvitationStatus Class

Format

An `R6Class` generator object

Methods

Public methods:

- `InvitationStatus$new()`
- `InvitationStatus$toJSON()`
- `InvitationStatus$fromJSON()`
- `InvitationStatus$toJSONString()`
- `InvitationStatus$fromJSONString()`
- `InvitationStatus$clone()`

Method `new()`**:**

Usage:

```
InvitationStatus$new(...)
```

Method `toJSON()`**:**

Usage:

```
InvitationStatus$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
InvitationStatus$fromJSON(InvitationStatusJson)
```

Method `toJSONString()`**:**

Usage:

```
InvitationStatus$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
InvitationStatus$fromJSONString(InvitationStatusJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
InvitationStatus$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`InvitationType` *InvitationType*

Description

InvitationType Class

Format

An `R6Class` generator object

Methods

Public methods:

- `InvitationType$new()`
- `InvitationType$toJSON()`
- `InvitationType$fromJSON()`
- `InvitationType$toJSONString()`
- `InvitationType$fromJSONString()`
- `InvitationType$clone()`

Method `new()`**:**

Usage:

```
InvitationType$new(...)
```

Method `toJSON()`**:**

Usage:

```
InvitationType$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
InvitationType$fromJSON(InvitationTypeJson)
```

Method `toJSONString()`**:**

Usage:

```
InvitationType$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
InvitationType$fromJSONString(InvitationTypeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
InvitationType$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`LastAccessedArray` *LastAccessedArray*

Description

LastAccessedArray Class

Format

An `R6Class` generator object

Public fields

`array_id` character [optional]

`array_name` character [optional]

`namespace` character [optional]

`accessed_time` numeric [optional]

`access_type` ActivityEventType [optional]

Methods

Public methods:

- LastAccessedArray$new()
- LastAccessedArray$toJSON()
- LastAccessedArray$fromJSON()
- LastAccessedArray$toJSONString()
- LastAccessedArray$fromJSONString()
- LastAccessedArray$clone()

Method new():

Usage:

```
LastAccessedArray$new(
  array_id = NULL,
  array_name = NULL,
  namespace = NULL,
  accessed_time = NULL,
  access_type = NULL,
  ...
)
```

Method toJSON():

Usage:

```
LastAccessedArray$toJSON()
```

Method fromJSON():

Usage:

```
LastAccessedArray$fromJSON(LastAccessedArrayJson)
```

Method toJSONString():

Usage:

```
LastAccessedArray$toJSONString()
```

Method fromJSONString():

Usage:

```
LastAccessedArray$fromJSONString(LastAccessedArrayJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
LastAccessedArray$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Layout *Layout*

Description

Layout Class

Format

An R6Class generator object

Methods

Public methods:

- Layout$new()
- Layout$toJSON()
- Layout$fromJSON()
- Layout$toJSONString()
- Layout$fromJSONString()
- Layout$clone()

Method `new()`**:**

Usage:

```
Layout$new(...)
```

Method `toJSON()`**:**

Usage:

```
Layout$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
Layout$fromJSON(LayoutJson)
```

Method `toJSONString()`**:**

Usage:

```
Layout$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Layout$fromJSONString(LayoutJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Layout$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`list_arrays` *Show listing of arrays*

Description

Returns a dataframe of metadata for all arrays that meet the filter applied.

Usage

```
list_arrays(
  public = FALSE,
  shared = FALSE,
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  file.type = NULL,
  exclude.file.type = NULL,
  file.property = NULL,
  ...
)
```

Arguments

`public`	logical TRUE means list public arrays
`shared`	logical TRUE means list shared arrays. If `public` and `shared` are both false then arrays owned by you are listed.
`page`	integer
`per.page`	integer
`search`	character
`namespace`	character
`orderby`	character
`permissions`	character
`tag`	list(character)
`exclude.tag`	list(character)
`file.type`	list(character)
`exclude.file.type`	list(character)
`file.property`	list(character)

Details

Note that this is a paginable API but default params return all results on one call, even hundreds of them. The `public` and `shared` arguments may not both be true.

Pagination information is set as an attribute of the returned data frame.

Value

Dataframe of metadata for all arrays in your account that meet the filter applied.

See Also

Other manual-layer functions: `array_info()`, `compute_sequentially()`, `compute()`, `delayed_args<-()`, `delayed_args()`, `delayed_array_udf()`, `delayed_generic_udf()`, `delayed_sql()`, `delayed()`, `deregister_array()`, `deregister_group()`, `deregister_udf()`, `execute_array_udf()`, `execute_generic_udf()`, `execute_multi_array_udf()`, `execute_sql_query()`, `get_udf_info()`, `group_info()`, `list_groups()`, `login()`, `register_array()`, `register_udf()`, `update_udf_info()`, `user_profile()`

---

list_groups *Show listing of groups*

Description

Returns a dataframe of metadata for all groups that meet the filter applied.

Usage

```
list_groups(
  public = FALSE,
  shared = FALSE,
  page = NULL,
  per.page = NULL,
  search = NULL,
  namespace = NULL,
  orderby = NULL,
  permissions = NULL,
  tag = NULL,
  exclude.tag = NULL,
  flat = FALSE,
  parent = NULL
)
```

Arguments

`public`	logical TRUE means list public arrays
`shared`	logical TRUE means list shared arrays. If `public` and `shared` are both false then arrays owned by you are listed.

page	integer
per.page	integer
search	character
namespace	character
orderby	character
permissions	character
tag	list(character)
exclude.tag	list(character)
flat	logical, if 'TRUE', ignores the nesting of groups and searches all of them
parent	character, search only the children of the groups with this uuid

Details

Note that this is a paginable API but default params return all results on one call, even hundreds of them.

Pagination information is set as an attribute of the returned data frame.

Value

A 'data.frame' of metadata for all groups in your account that meet the filter applied.

See Also

Other manual-layer functions: `array_info()`, `compute_sequentially()`, `compute()`, `delayed_args<-()`, `delayed_args()`, `delayed_array_udf()`, `delayed_generic_udf()`, `delayed_sql()`, `delayed()`, `deregister_array()`, `deregister_group()`, `deregister_udf()`, `execute_array_udf()`, `execute_generic_udf()`, `execute_multi_array_udf()`, `execute_sql_query()`, `get_udf_info()`, `group_info()`, `list_arrays()`, `login()`, `register_array()`, `register_udf()`, `update_udf_info()`, `user_profile()`

---

login *Log in to TileDB Cloud*

Description

This function can be used to override the default setup made at package load.

Usage

```
login(
  username,
  password,
  api_key,
  host,
  remember_me = TRUE,
  write_config = FALSE
)
```

Arguments

username	A character value with the username, if present password is also needed.
password	A character value with the password, if present username is also needed.
api_key	A character value with the access token, it can be used instead of username and password.
host	A character value with remote host to connect to.
remember_me	A boolean to select a session with for 24 hours instead of 8 hours, used only when a new session is requested.
write_config	A boolean to write the login information to `~/.tiledb/cloud.json` from where it can be read for subsequent sessions. This is only done when requested by this parameter, which is `FALSE` by default.

Details

It can operate in two modes. Either a username and a password are supplied as environment variable `TILEDB_REST_USERNAME` and `TILEDB_REST_PASSWORD`. As an alternative, an access token can be supplied via `TILEDB_REST_TOKEN`. The values are used to instantiate a new API client object. If no token was supplied, a new session is requested and the token assigned to that session is used.

Function arguments are optional, and can be used to override the default configuration values obtained by `configure` from either the environment variables or the configuration file.

Value

Nothing is returned; the function is called for a side effect of storing the values in the package environment.

See Also

Other manual-layer functions: `array_info()`, `compute_sequentially()`, `compute()`, `delayed_args<-()`, `delayed_args()`, `delayed_array_udf()`, `delayed_generic_udf()`, `delayed_sql()`, `delayed()`, `deregister_array()`, `deregister_group()`, `deregister_udf()`, `execute_array_udf()`, `execute_generic_udf()`, `execute_multi_array_udf()`, `execute_sql_query()`, `get_udf_info()`, `group_info()`, `list_arrays()`, `list_groups()`, `register_array()`, `register_udf()`, `update_udf_info()`, `user_profile()`

MaxBufferSizes *MaxBufferSizes*

Description

MaxBufferSizes Class

Format

An `R6Class` generator object

Public fields

`maxBufferSizes` list(AttributeBufferSize) [optional]

Methods

Public methods:

- `MaxBufferSizes$new()`
- `MaxBufferSizes$toJSON()`
- `MaxBufferSizes$fromJSON()`
- `MaxBufferSizes$toJSONString()`
- `MaxBufferSizes$fromJSONString()`
- `MaxBufferSizes$clone()`

Method `new()`:

Usage:

```
MaxBufferSizes$new(maxBufferSizes = NULL, ...)
```

Method `toJSON()`:

Usage:

```
MaxBufferSizes$toJSON()
```

Method `fromJSON()`:

Usage:

```
MaxBufferSizes$fromJSON(MaxBufferSizesJson)
```

Method `toJSONString()`:

Usage:

```
MaxBufferSizes$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
MaxBufferSizes$fromJSONString(MaxBufferSizesJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
MaxBufferSizes$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

MLModelFavorite *MLModelFavorite*

Description

MLModelFavorite Class

Format

An R6Class generator object

Public fields

mlmodel_uuid character [optional]

namespace character [optional]

name character [optional]

Methods

Public methods:

- MLModelFavorite$new()
- MLModelFavorite$toJSON()
- MLModelFavorite$fromJSON()
- MLModelFavorite$toJSONString()
- MLModelFavorite$fromJSONString()
- MLModelFavorite$clone()

Method new():

Usage:

```
MLModelFavorite$new(mlmodel_uuid = NULL, namespace = NULL, name = NULL, ...)
```

Method toJSON():

Usage:

```
MLModelFavorite$toJSON()
```

Method fromJSON():

Usage:

```
MLModelFavorite$fromJSON(MLModelFavoriteJson)
```

Method toJSONString():

Usage:

```
MLModelFavorite$toJSONString()
```

Method fromJSONString():

Usage:

```
MLModelFavorite$fromJSONString(MLModelFavoriteJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
MLModelFavorite$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`MLModelFavoritesData` *MLModelFavoritesData*

Description

MLModelFavoritesData Class

Format

An `R6Class` generator object

Public fields

`mlmodels` list(ArrayInfo) [optional]

`pagination_metadata` PaginationMetadata [optional]

Methods

Public methods:

- `MLModelFavoritesData$new()`
- `MLModelFavoritesData$toJSON()`
- `MLModelFavoritesData$fromJSON()`
- `MLModelFavoritesData$toJSONString()`
- `MLModelFavoritesData$fromJSONString()`
- `MLModelFavoritesData$clone()`

Method `new()`**:**

Usage:

```
MLModelFavoritesData$new(mlmodels = NULL, pagination_metadata = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
MLModelFavoritesData$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
MLModelFavoritesData$fromJSON(MLModelFavoritesDataJson)
```

Method `toJSONString()`:

Usage:

```
MLModelFavoritesData$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
MLModelFavoritesData$fromJSONString(MLModelFavoritesDataJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
MLModelFavoritesData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`MultiArrayUDF` *MultiArrayUDF*

Description

MultiArrayUDF Class

Format

An `R6Class` generator object

Public fields

`udf_info_name` character [optional]

`language` UDFLanguage [optional]

`version` character [optional]

`image_name` character [optional]

`resource_class` character [optional]

`exec` character [optional]

`exec_raw` character [optional]

`result_format` ResultFormat [optional]

`task_name` character [optional]

`argument` character [optional]

`arguments_json` list(TGUDFArgument) [optional]

`stored_param_uuids` list(character) [optional]

`store_results` character [optional]

`dont_download_results` character [optional]

`ranges` QueryRanges [optional]

`subarray` UDFSubarray [optional]

`buffers` list(character) [optional]

`arrays` list(UDFArrayDetails) [optional]

`timeout` integer [optional]

`task_graph_uuid` character [optional]

`client_node_uuid` character [optional]

Methods

Public methods:

- `MultiArrayUDF$new()`
- `MultiArrayUDF$toJSON()`
- `MultiArrayUDF$fromJSON()`
- `MultiArrayUDF$toJSONString()`
- `MultiArrayUDF$fromJSONString()`
- `MultiArrayUDF$clone()`

Method `new()`:

Usage:

```
MultiArrayUDF$new(
  udf_info_name = NULL,
  language = NULL,
  version = NULL,
  image_name = NULL,
  resource_class = NULL,
  exec = NULL,
  exec_raw = NULL,
  result_format = NULL,
  task_name = NULL,
  argument = NULL,
  arguments_json = NULL,
  stored_param_uuids = NULL,
  store_results = NULL,
  dont_download_results = NULL,
  ranges = NULL,
  subarray = NULL,
  buffers = NULL,
  arrays = NULL,
  timeout = NULL,
  task_graph_uuid = NULL,
  client_node_uuid = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
MultiArrayUDF$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
MultiArrayUDF$fromJSON(MultiArrayUDFJson)
```

Method `toJSONString()`**:**

Usage:

```
MultiArrayUDF$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
MultiArrayUDF$fromJSONString(MultiArrayUDFJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
MultiArrayUDF$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`NamespaceActions` *NamespaceActions*

Description

NamespaceActions Class

Format

An `R6Class` generator object

Methods

Public methods:

- `NamespaceActions$new()`
- `NamespaceActions$toJSON()`
- `NamespaceActions$fromJSON()`
- `NamespaceActions$toJSONString()`
- `NamespaceActions$fromJSONString()`
- `NamespaceActions$clone()`

Method `new()`**:**

Usage:

```
NamespaceActions$new(...)
```

Method `toJSON()`**:**

Usage:

```
NamespaceActions$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
NamespaceActions$fromJSON(NamespaceActionsJson)
```

Method `toJSONString()`**:**

Usage:

```
NamespaceActions$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
NamespaceActions$fromJSONString(NamespaceActionsJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NamespaceActions$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`NonEmptyDomain` *NonEmptyDomain*

Description

NonEmptyDomain Class

Format

An `R6Class` generator object

Public fields

`nonEmptyDomain` DomainArray

`isEmpty` character

Methods

Public methods:

- NonEmptyDomain$new()
- NonEmptyDomain$toJSON()
- NonEmptyDomain$fromJSON()
- NonEmptyDomain$toJSONString()
- NonEmptyDomain$fromJSONString()
- NonEmptyDomain$clone()

Method `new()`:

Usage:

```
NonEmptyDomain$new(nonEmptyDomain, isEmpty, ...)
```

Method `toJSON()`:

Usage:

```
NonEmptyDomain$toJSON()
```

Method `fromJSON()`:

Usage:

```
NonEmptyDomain$fromJSON(NonEmptyDomainJson)
```

Method `toJSONString()`:

Usage:

```
NonEmptyDomain$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
NonEmptyDomain$fromJSONString(NonEmptyDomainJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
NonEmptyDomain$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`NotebookApi` *Notebook operations*

Description

tiledbcloud.Notebook

Format

An `R6Class` generator object

Methods

GetNotebookServerStatus Get status of the notebook server

@*param* namespace character

- @*returnType* NotebookStatus

- status code : 200 | status of running notebook
- return type : NotebookStatus
- response headers :

- status code : 202 | Notebook server is pending
- response headers :

- status code : 402 | Payment required
- return type : Error
- response headers :

- status code : 404 | Notebook is not running
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ShutdownNotebookServer Shutdown a notebook server

@*param* namespace character

- status code : 204 | Notebook shutdown successfully

- response headers :

- status code : 404 | Notebook is not running
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

UpdateNotebookName update name on a notebok, moving related S3 object to new location

@*param* namespace character

- @*param* array character
- @*param* notebook.metadata ArrayInfoUpdate
- status code : 204 | notebook name updated successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `NotebookApi$new()`
- `NotebookApi$GetNotebookServerStatus()`
- `NotebookApi$GetNotebookServerStatusWithHttpInfo()`
- `NotebookApi$ShutdownNotebookServer()`
- `NotebookApi$ShutdownNotebookServerWithHttpInfo()`
- `NotebookApi$UpdateNotebookName()`
- `NotebookApi$UpdateNotebookNameWithHttpInfo()`
- `NotebookApi$clone()`

Method `new()`:

Usage:

```
NotebookApi$new(apiClient)
```

Method `GetNotebookServerStatus()`:

Usage:

```
NotebookApi$GetNotebookServerStatus(namespace, ...)
```

Method `GetNotebookServerStatusWithHttpInfo()`**:**

Usage:

```
NotebookApi$GetNotebookServerStatusWithHttpInfo(namespace, ...)
```

Method `ShutdownNotebookServer()`**:**

Usage:

```
NotebookApi$ShutdownNotebookServer(namespace, ...)
```

Method `ShutdownNotebookServerWithHttpInfo()`**:**

Usage:

```
NotebookApi$ShutdownNotebookServerWithHttpInfo(namespace, ...)
```

Method `UpdateNotebookName()`**:**

Usage:

```
NotebookApi$UpdateNotebookName(namespace, array, notebook.metadata, ...)
```

Method `UpdateNotebookNameWithHttpInfo()`**:**

Usage:

```
NotebookApi$UpdateNotebookNameWithHttpInfo(
  namespace,
  array,
  notebook.metadata,
  ...
)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NotebookApi$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
####################  GetNotebookServerStatus  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace notebook is in (an organization name or user's usernam

api.instance <- NotebookApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';
```

```
#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetNotebookServerStatus(var.namespace)


####################  ShutdownNotebookServer  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace notebook is in (an organization name or user's usernam

api.instance <- NotebookApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ShutdownNotebookServer(var.namespace)


####################  UpdateNotebookName  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of notebook (array) that is url-encoded
var.notebook.metadata <- ArrayInfoUpdate$new() # ArrayInfoUpdate | notebook (array) metadata to update

api.instance <- NotebookApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$UpdateNotebookName(var.namespace, var.array, var.notebook.metadata)



## End(Not run)
```

NotebookFavorite *NotebookFavorite*

Description

NotebookFavorite Class

Format

An R6Class generator object

Public fields

notebook_uuid character [optional]

namespace character [optional]

name character [optional]

Methods

Public methods:

- NotebookFavorite$new()
- NotebookFavorite$toJSON()
- NotebookFavorite$fromJSON()
- NotebookFavorite$toJSONString()
- NotebookFavorite$fromJSONString()
- NotebookFavorite$clone()

Method new():

Usage:

```
NotebookFavorite$new(notebook_uuid = NULL, namespace = NULL, name = NULL, ...)
```

Method toJSON():

Usage:

```
NotebookFavorite$toJSON()
```

Method fromJSON():

Usage:

```
NotebookFavorite$fromJSON(NotebookFavoriteJson)
```

Method toJSONString():

Usage:

```
NotebookFavorite$toJSONString()
```

Method fromJSONString():

Usage:

```
NotebookFavorite$fromJSONString(NotebookFavoriteJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NotebookFavorite$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`NotebookFavoritesData` *NotebookFavoritesData*

Description

NotebookFavoritesData Class

Format

An `R6Class` generator object

Public fields

`notebooks` list(ArrayInfo) [optional]

`pagination_metadata` PaginationMetadata [optional]

Methods

Public methods:

- `NotebookFavoritesData$new()`
- `NotebookFavoritesData$toJSON()`
- `NotebookFavoritesData$fromJSON()`
- `NotebookFavoritesData$toJSONString()`
- `NotebookFavoritesData$fromJSONString()`
- `NotebookFavoritesData$clone()`

Method `new()`**:**

Usage:

```
NotebookFavoritesData$new(notebooks = NULL, pagination_metadata = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
NotebookFavoritesData$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
NotebookFavoritesData$fromJSON(NotebookFavoritesDataJson)
```

Method `toJSONString()`:

Usage:

```
NotebookFavoritesData$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
NotebookFavoritesData$fromJSONString(NotebookFavoritesDataJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
NotebookFavoritesData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`NotebooksApi` *Notebooks operations*

Description

tiledbcloud.Notebooks

Format

An `R6Class` generator object

Methods

NotebooksNamespaceArrayEndTimestampsGet retrieve a list of timestamps from the array fragment info listing in milliseconds, paginated

@*param* namespace character

- @*param* array character
- @*param* page integer
- @*param* per.page integer
- @*returnType* ArrayEndTimestampData

- status code : 200 | list of timestamps in milliseconds, paginated
- return type : ArrayEndTimestampData
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `NotebooksApi$new()`
- `NotebooksApi$NotebooksNamespaceArrayEndTimestampsGet()`
- `NotebooksApi$NotebooksNamespaceArrayEndTimestampsGetWithHttpInfo()`
- `NotebooksApi$clone()`

Method `new()`:

Usage:

```
NotebooksApi$new(apiClient)
```

Method `NotebooksNamespaceArrayEndTimestampsGet()`:

Usage:

```
NotebooksApi$NotebooksNamespaceArrayEndTimestampsGet(
  namespace,
  array,
  page = NULL,
  per.page = NULL,
  ...
)
```

Method `NotebooksNamespaceArrayEndTimestampsGetWithHttpInfo()`:

Usage:

```
NotebooksApi$NotebooksNamespaceArrayEndTimestampsGetWithHttpInfo(
  namespace,
  array,
  page = NULL,
  per.page = NULL,
  ...
)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
NotebooksApi$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
####################  NotebooksNamespaceArrayEndTimestampsGet  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit

api.instance <- NotebooksApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$NotebooksNamespaceArrayEndTimestampsGet(var.namespace, var.array, page=var.page, per.page



## End(Not run)
```

`NotebookStatus` *NotebookStatus*

Description

NotebookStatus Class

Format

An `R6Class` generator object

Public fields

`namespace` character [optional]

`uptime` integer [optional]

`cpu_usage` integer [optional]

`memory_usage` integer [optional]

`memory_limit` integer [optional]

`cpu_count` integer [optional]

Methods

Public methods:

- NotebookStatus$new()
- NotebookStatus$toJSON()
- NotebookStatus$fromJSON()
- NotebookStatus$toJSONString()
- NotebookStatus$fromJSONString()
- NotebookStatus$clone()

Method `new()`:

Usage:

```
NotebookStatus$new(
  namespace = NULL,
  uptime = NULL,
  cpu_usage = NULL,
  memory_usage = NULL,
  memory_limit = NULL,
  cpu_count = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
NotebookStatus$toJSON()
```

Method `fromJSON()`:

Usage:

```
NotebookStatus$fromJSON(NotebookStatusJson)
```

Method `toJSONString()`:

Usage:

```
NotebookStatus$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
NotebookStatus$fromJSONString(NotebookStatusJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
NotebookStatus$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Organization` *Organization*

Description

Organization Class

Format

An `R6Class` generator object

Public fields

`id` character [optional]

`role` OrganizationRoles [optional]

`name` character

`created_at` character [optional]

`updated_at` character [optional]

`logo` character [optional]

`description` character [optional]

`users` list(OrganizationUser) [optional]

`allowed_actions` list(NamespaceActions) [optional]

`num_of_arrays` numeric [optional]

`enabled_features` list(character) [optional]

`unpaid_subscription` character [optional]

`default_s3_path` character [optional]

`default_s3_path_credentials_name` character [optional]

`stripe_connect` character [optional]

Methods

Public methods:

- `Organization$new()`
- `Organization$toJSON()`
- `Organization$fromJSON()`
- `Organization$toJSONString()`
- `Organization$fromJSONString()`
- `Organization$clone()`

Method `new()`:

Usage:

```
Organization$new(
  name,
  id = NULL,
  role = NULL,
  created_at = NULL,
  updated_at = NULL,
  logo = NULL,
  description = NULL,
  users = NULL,
  allowed_actions = NULL,
  num_of_arrays = NULL,
  enabled_features = NULL,
  unpaid_subscription = NULL,
  default_s3_path = NULL,
  default_s3_path_credentials_name = NULL,
  stripe_connect = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
Organization$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
Organization$fromJSON(OrganizationJson)
```

Method `toJSONString()`**:**

Usage:

```
Organization$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Organization$fromJSONString(OrganizationJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Organization$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`OrganizationApi` *Organization operations*

Description

tiledbcloud.Organization

Format

An `R6Class` generator object

Methods

AddAWSAccessCredentials Add aws keys

@*param* namespace character

- @*param* aws.access.credentials AWSAccessCredentials
- status code : 204 | AWS keys added successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

AddUserToOrganization add a user to an organization

@*param* organization character

- @*param* user OrganizationUser
- status code : 204 | user added to organization successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

CheckAWSAccessCredentials Check if aws keys are set

@*param* namespace character

- @*returnType* list(AWSAccessCredentials)

- status code : 200 | AWS keys are set
- return type : array[AWSAccessCredentials]

- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

CheckAWSAccessCredentialsByName Check if aws keys are set by name

@*param* namespace character
- @*param* name character
- @*returnType* AWSAccessCredentials

- status code : 200 | AWS keys are set
- return type : AWSAccessCredentials
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

CreateOrganization create a organization, the user creating will be listed as owner

@*param* organization Organization
- status code : 204 | organization created successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

DeleteAWSAccessCredentials delete a AWS Access credentials in a namespace. This will likely cause arrays to become unreachable

@*param* namespace character
- @*param* name character
- status code : 204 | AWS credentials deleted
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

DeleteOrganization delete a organization

@*param* organization character

- status code : 204 | organization deleted
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

DeleteUserFromOrganization delete a user from an organization

@*param* organization character

- @*param* username character
- status code : 204 | user delete from organization successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetAllOrganizations get all organizations that the user is member of

@*returnType* list(Organization)

- status code : 200 | array of organizations the user is member of
- return type : array[Organization]
- response headers :

- status code : 400 | Error finding organizations
- response headers :

- status code : 500 | Request user not found, or has empty context
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

GetOrganization get a organization

@*param* organization character

- @*returnType* Organization

- status code : 200 | organization details
- return type : Organization
- response headers :

- status code : 404 | Organization does not exist
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetOrganizationUser get a user from an organization

@*param* organization character

- @*param* username character
- @*returnType* OrganizationUser

- status code : 200 | user from organization
- return type : OrganizationUser
- response headers :

- status code : 404 | User is not in organization
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

UpdateAWSAccessCredentials Update aws keys or associated buckets. This will update the key associations for each array in the namespace

@*param* namespace character

- @*param* name character
- @*param* aws.access.credentials AWSAccessCredentials
- status code : 204 | AWS keys updated successfully

- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

UpdateOrganization update a organization

@*param* organization character

- @*param* organization.details Organization
- status code : 204 | organization updated successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

UpdateUserInOrganization update a user in an organization

@*param* organization character

- @*param* username character
- @*param* user OrganizationUser
- status code : 204 | user update in organization successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `OrganizationApi$new()`
- `OrganizationApi$AddAWSAccessCredentials()`
- `OrganizationApi$AddAWSAccessCredentialsWithHttpInfo()`
- `OrganizationApi$AddUserToOrganization()`
- `OrganizationApi$AddUserToOrganizationWithHttpInfo()`
- `OrganizationApi$CheckAWSAccessCredentials()`

- OrganizationApi$CheckAWSAccessCredentialsWithHttpInfo()
- OrganizationApi$CheckAWSAccessCredentialsByName()
- OrganizationApi$CheckAWSAccessCredentialsByNameWithHttpInfo()
- OrganizationApi$CreateOrganization()
- OrganizationApi$CreateOrganizationWithHttpInfo()
- OrganizationApi$DeleteAWSAccessCredentials()
- OrganizationApi$DeleteAWSAccessCredentialsWithHttpInfo()
- OrganizationApi$DeleteOrganization()
- OrganizationApi$DeleteOrganizationWithHttpInfo()
- OrganizationApi$DeleteUserFromOrganization()
- OrganizationApi$DeleteUserFromOrganizationWithHttpInfo()
- OrganizationApi$GetAllOrganizations()
- OrganizationApi$GetAllOrganizationsWithHttpInfo()
- OrganizationApi$GetOrganization()
- OrganizationApi$GetOrganizationWithHttpInfo()
- OrganizationApi$GetOrganizationUser()
- OrganizationApi$GetOrganizationUserWithHttpInfo()
- OrganizationApi$UpdateAWSAccessCredentials()
- OrganizationApi$UpdateAWSAccessCredentialsWithHttpInfo()
- OrganizationApi$UpdateOrganization()
- OrganizationApi$UpdateOrganizationWithHttpInfo()
- OrganizationApi$UpdateUserInOrganization()
- OrganizationApi$UpdateUserInOrganizationWithHttpInfo()
- OrganizationApi$clone()

Method `new()`**:**

Usage:

```
OrganizationApi$new(apiClient)
```

Method `AddAWSAccessCredentials()`**:**

Usage:

```
OrganizationApi$AddAWSAccessCredentials(namespace, aws.access.credentials, ...)
```

Method `AddAWSAccessCredentialsWithHttpInfo()`**:**

Usage:

```
OrganizationApi$AddAWSAccessCredentialsWithHttpInfo(
  namespace,
  aws.access.credentials,
  ...
)
```

Method `AddUserToOrganization()`**:**

Usage:

```
OrganizationApi$AddUserToOrganization(organization, user, ...)
```

Method `AddUserToOrganizationWithHttpInfo()`**:**

Usage:

```
OrganizationApi$AddUserToOrganizationWithHttpInfo(organization, user, ...)
```

Method `CheckAWSAccessCredentials()`**:**

Usage:

```
OrganizationApi$CheckAWSAccessCredentials(namespace, ...)
```

Method `CheckAWSAccessCredentialsWithHttpInfo()`**:**

Usage:

```
OrganizationApi$CheckAWSAccessCredentialsWithHttpInfo(namespace, ...)
```

Method `CheckAWSAccessCredentialsByName()`**:**

Usage:

```
OrganizationApi$CheckAWSAccessCredentialsByName(namespace, name, ...)
```

Method `CheckAWSAccessCredentialsByNameWithHttpInfo()`**:**

Usage:

```
OrganizationApi$CheckAWSAccessCredentialsByNameWithHttpInfo(
  namespace,
  name,
  ...
)
```

Method `CreateOrganization()`**:**

Usage:

```
OrganizationApi$CreateOrganization(organization, ...)
```

Method `CreateOrganizationWithHttpInfo()`**:**

Usage:

```
OrganizationApi$CreateOrganizationWithHttpInfo(organization, ...)
```

Method `DeleteAWSAccessCredentials()`**:**

Usage:

```
OrganizationApi$DeleteAWSAccessCredentials(namespace, name, ...)
```

Method `DeleteAWSAccessCredentialsWithHttpInfo()`**:**

Usage:

```
OrganizationApi$DeleteAWSAccessCredentialsWithHttpInfo(namespace, name, ...)
```

Method `DeleteOrganization()`**:**

Usage:

```
OrganizationApi$DeleteOrganization(organization, ...)
```

Method `DeleteOrganizationWithHttpInfo()`**:**

Usage:

```
OrganizationApi$DeleteOrganizationWithHttpInfo(organization, ...)
```

Method `DeleteUserFromOrganization()`**:**

Usage:

```
OrganizationApi$DeleteUserFromOrganization(organization, username, ...)
```

Method `DeleteUserFromOrganizationWithHttpInfo()`**:**

Usage:

```
OrganizationApi$DeleteUserFromOrganizationWithHttpInfo(
  organization,
  username,
  ...
)
```

Method `GetAllOrganizations()`**:**

Usage:

```
OrganizationApi$GetAllOrganizations(...)
```

Method `GetAllOrganizationsWithHttpInfo()`**:**

Usage:

```
OrganizationApi$GetAllOrganizationsWithHttpInfo(...)
```

Method `GetOrganization()`**:**

Usage:

```
OrganizationApi$GetOrganization(organization, ...)
```

Method `GetOrganizationWithHttpInfo()`**:**

Usage:

```
OrganizationApi$GetOrganizationWithHttpInfo(organization, ...)
```

Method `GetOrganizationUser()`**:**

Usage:

```
OrganizationApi$GetOrganizationUser(organization, username, ...)
```

Method `GetOrganizationUserWithHttpInfo()`**:**

Usage:

```
OrganizationApi$GetOrganizationUserWithHttpInfo(organization, username, ...)
```

Method `UpdateAWSAccessCredentials()`**:**

Usage:

```
OrganizationApi$UpdateAWSAccessCredentials(
  namespace,
  name,
  aws.access.credentials,
  ...
)
```

Method `UpdateAWSAccessCredentialsWithHttpInfo()`**:**

Usage:

```
OrganizationApi$UpdateAWSAccessCredentialsWithHttpInfo(
  namespace,
  name,
  aws.access.credentials,
  ...
)
```

Method `UpdateOrganization()`**:**

Usage:

```
OrganizationApi$UpdateOrganization(organization, organization.details, ...)
```

Method `UpdateOrganizationWithHttpInfo()`**:**

Usage:

```
OrganizationApi$UpdateOrganizationWithHttpInfo(
  organization,
  organization.details,
  ...
)
```

Method `UpdateUserInOrganization()`**:**

Usage:

```
OrganizationApi$UpdateUserInOrganization(organization, username, user, ...)
```

Method `UpdateUserInOrganizationWithHttpInfo()`**:**

Usage:

```
OrganizationApi$UpdateUserInOrganizationWithHttpInfo(
  organization,
  username,
  user,
  ...
)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
OrganizationApi$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
####################  AddAWSAccessCredentials  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace
```

```
var.aws.access.credentials <- AWSAccessCredentials$new() # AWSAccessCredentials | aws access credentials to store

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$AddAWSAccessCredentials(var.namespace, var.aws.access.credentials)


####################  AddUserToOrganization  ####################

library(tiledbcloud)
var.organization <- 'organization_example' # character | organization name
var.user <- OrganizationUser$new() # OrganizationUser | user to add

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$AddUserToOrganization(var.organization, var.user)


####################  CheckAWSAccessCredentials  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$CheckAWSAccessCredentials(var.namespace)
```

```
####################  CheckAWSAccessCredentialsByName  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace
var.name <- 'name_example' # character | name

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$CheckAWSAccessCredentialsByName(var.namespace, var.name)


####################  CreateOrganization  ####################

library(tiledbcloud)
var.organization <- Organization$new() # Organization | organization to create

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$CreateOrganization(var.organization)


####################  DeleteAWSAccessCredentials  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace
var.name <- 'name_example' # character | name

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
```

```
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteAWSAccessCredentials(var.namespace, var.name)


####################  DeleteOrganization  ####################

library(tiledbcloud)
var.organization <- 'organization_example' # character | organization name or ID

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteOrganization(var.organization)


####################  DeleteUserFromOrganization  ####################

library(tiledbcloud)
var.organization <- 'organization_example' # character | organization name
var.username <- 'username_example' # character | username to manipulate

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteUserFromOrganization(var.organization, var.username)


####################  GetAllOrganizations  ####################

library(tiledbcloud)

api.instance <- OrganizationApi$new()
```

```
#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetAllOrganizations()


####################  GetOrganization  ####################

library(tiledbcloud)
var.organization <- 'organization_example' # character | organization name or ID

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetOrganization(var.organization)


####################  GetOrganizationUser  ####################

library(tiledbcloud)
var.organization <- 'organization_example' # character | organization name
var.username <- 'username_example' # character | username to manipulate

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetOrganizationUser(var.organization, var.username)


####################  UpdateAWSAccessCredentials  ####################
```

```
library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace
var.name <- 'name_example' # character | name
var.aws.access.credentials <- AWSAccessCredentials$new() # AWSAccessCredentials | aws credentials to update

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$UpdateAWSAccessCredentials(var.namespace, var.name, var.aws.access.credentials)


####################  UpdateOrganization  ####################

library(tiledbcloud)
var.organization <- 'organization_example' # character | organization name or ID
var.organization.details <- Organization$new() # Organization | organization details to update

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$UpdateOrganization(var.organization, var.organization.details)


####################  UpdateUserInOrganization  ####################

library(tiledbcloud)
var.organization <- 'organization_example' # character | organization name
var.username <- 'username_example' # character | username to manipulate
var.user <- OrganizationUser$new() # OrganizationUser | user details to update

api.instance <- OrganizationApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
```

```
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$UpdateUserInOrganization(var.organization, var.username, var.user)



## End(Not run)
```

`OrganizationRoles` *OrganizationRoles*

Description

OrganizationRoles Class

Format

An `R6Class` generator object

Methods

Public methods:

- `OrganizationRoles$new()`
- `OrganizationRoles$toJSON()`
- `OrganizationRoles$fromJSON()`
- `OrganizationRoles$toJSONString()`
- `OrganizationRoles$fromJSONString()`
- `OrganizationRoles$clone()`

Method `new()`:

Usage:

```
OrganizationRoles$new(...)
```

Method `toJSON()`:

Usage:

```
OrganizationRoles$toJSON()
```

Method `fromJSON()`:

Usage:

```
OrganizationRoles$fromJSON(OrganizationRolesJson)
```

Method `toJSONString()`:

Usage:

```
OrganizationRoles$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
OrganizationRoles$fromJSONString(OrganizationRolesJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
OrganizationRoles$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`OrganizationUser` *OrganizationUser*

Description

OrganizationUser Class

Format

An `R6Class` generator object

Public fields

`user_id` character [optional]

`organization_id` character [optional]

`username` character [optional]

`organization_name` character [optional]

`role` OrganizationRoles [optional]

`allowed_actions` list(NamespaceActions) [optional]

Methods

Public methods:

- `OrganizationUser$new()`
- `OrganizationUser$toJSON()`
- `OrganizationUser$fromJSON()`
- `OrganizationUser$toJSONString()`
- `OrganizationUser$fromJSONString()`
- `OrganizationUser$clone()`

Method `new()`**:**

Usage:

```
OrganizationUser$new(
  user_id = NULL,
  organization_id = NULL,
  username = NULL,
  organization_name = NULL,
  role = NULL,
  allowed_actions = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
OrganizationUser$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
OrganizationUser$fromJSON(OrganizationUserJson)
```

Method `toJSONString()`**:**

Usage:

```
OrganizationUser$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
OrganizationUser$fromJSONString(OrganizationUserJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
OrganizationUser$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`PaginationMetadata` *PaginationMetadata*

Description

PaginationMetadata Class

Format

An `R6Class` generator object

Public fields

`page` numeric [optional]

`per_page` numeric [optional]

`total_pages` numeric [optional]

`total_items` numeric [optional]

Methods

Public methods:

- `PaginationMetadata$new()`
- `PaginationMetadata$toJSON()`
- `PaginationMetadata$fromJSON()`
- `PaginationMetadata$toJSONString()`
- `PaginationMetadata$fromJSONString()`
- `PaginationMetadata$clone()`

Method `new()`:

Usage:

```
PaginationMetadata$new(
  page = NULL,
  per_page = NULL,
  total_pages = NULL,
  total_items = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
PaginationMetadata$toJSON()
```

Method `fromJSON()`:

Usage:

```
PaginationMetadata$fromJSON(PaginationMetadataJson)
```

Method `toJSONString()`:

Usage:

```
PaginationMetadata$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
PaginationMetadata$fromJSONString(PaginationMetadataJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
PaginationMetadata$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Pricing` *Pricing*

Description

Pricing Class

Format

An `R6Class` generator object

Public fields

`id` character [optional]

`array_uuid` character [optional]

`pricing_name` character [optional]

`pricing_type` PricingType [optional]

`product_name` character [optional]

`product_statement_descriptor` character [optional]

`product_unit_label` PricingUnitLabel [optional]

`currency` PricingCurrency [optional]

`aggregate_usage` PricingAggregateUsage [optional]

`interval` PricingInterval [optional]

`divided_by` integer [optional]

`charge` numeric [optional]

`activated` character [optional]

Methods

Public methods:

- `Pricing$new()`
- `Pricing$toJSON()`
- `Pricing$fromJSON()`
- `Pricing$toJSONString()`
- `Pricing$fromJSONString()`
- `Pricing$clone()`

Method `new()`:

Usage:

```
Pricing$new(
  id = NULL,
  array_uuid = NULL,
  pricing_name = NULL,
  pricing_type = NULL,
  product_name = NULL,
  product_statement_descriptor = NULL,
  product_unit_label = NULL,
  currency = NULL,
  aggregate_usage = NULL,
  interval = NULL,
  divided_by = NULL,
  charge = NULL,
  activated = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
Pricing$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
Pricing$fromJSON(PricingJson)
```

Method `toJSONString()`**:**

Usage:

```
Pricing$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Pricing$fromJSONString(PricingJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Pricing$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

PricingAggregateUsage *PricingAggregateUsage*

Description

PricingAggregateUsage Class

Format

An R6Class generator object

Methods

Public methods:

- PricingAggregateUsage$new()
- PricingAggregateUsage$toJSON()
- PricingAggregateUsage$fromJSON()
- PricingAggregateUsage$toJSONString()
- PricingAggregateUsage$fromJSONString()
- PricingAggregateUsage$clone()

Method `new()`**:**

Usage:

```
PricingAggregateUsage$new(...)
```

Method `toJSON()`**:**

Usage:

```
PricingAggregateUsage$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
PricingAggregateUsage$fromJSON(PricingAggregateUsageJson)
```

Method `toJSONString()`**:**

Usage:

```
PricingAggregateUsage$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
PricingAggregateUsage$fromJSONString(PricingAggregateUsageJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PricingAggregateUsage$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

PricingCurrency *PricingCurrency*

Description

PricingCurrency Class

Format

An R6Class generator object

Methods

Public methods:

- PricingCurrency$new()
- PricingCurrency$toJSON()
- PricingCurrency$fromJSON()
- PricingCurrency$toJSONString()
- PricingCurrency$fromJSONString()
- PricingCurrency$clone()

Method `new()`**:**

Usage:

```
PricingCurrency$new(...)
```

Method `toJSON()`**:**

Usage:

```
PricingCurrency$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
PricingCurrency$fromJSON(PricingCurrencyJson)
```

Method `toJSONString()`**:**

Usage:

```
PricingCurrency$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
PricingCurrency$fromJSONString(PricingCurrencyJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PricingCurrency$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

PricingInterval *PricingInterval*

Description

PricingInterval Class

Format

An R6Class generator object

Methods

Public methods:

- PricingInterval$new()
- PricingInterval$toJSON()
- PricingInterval$fromJSON()
- PricingInterval$toJSONString()
- PricingInterval$fromJSONString()
- PricingInterval$clone()

Method new():

Usage:

```
PricingInterval$new(...)
```

Method toJSON():

Usage:

```
PricingInterval$toJSON()
```

Method fromJSON():

Usage:

```
PricingInterval$fromJSON(PricingIntervalJson)
```

Method toJSONString():

Usage:

```
PricingInterval$toJSONString()
```

Method fromJSONString():

Usage:

```
PricingInterval$fromJSONString(PricingIntervalJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
PricingInterval$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

PricingType *PricingType*

Description

PricingType Class

Format

An R6Class generator object

Methods

Public methods:

- PricingType$new()
- PricingType$toJSON()
- PricingType$fromJSON()
- PricingType$toJSONString()
- PricingType$fromJSONString()
- PricingType$clone()

Method `new()`**:**

Usage:

```
PricingType$new(...)
```

Method `toJSON()`**:**

Usage:

```
PricingType$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
PricingType$fromJSON(PricingTypeJson)
```

Method `toJSONString()`**:**

Usage:

```
PricingType$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
PricingType$fromJSONString(PricingTypeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PricingType$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

PricingUnitLabel *PricingUnitLabel*

Description

PricingUnitLabel Class

Format

An R6Class generator object

Methods

Public methods:

- PricingUnitLabel$new()
- PricingUnitLabel$toJSON()
- PricingUnitLabel$fromJSON()
- PricingUnitLabel$toJSONString()
- PricingUnitLabel$fromJSONString()
- PricingUnitLabel$clone()

Method new():

Usage:

```
PricingUnitLabel$new(...)
```

Method toJSON():

Usage:

```
PricingUnitLabel$toJSON()
```

Method fromJSON():

Usage:

```
PricingUnitLabel$fromJSON(PricingUnitLabelJson)
```

Method toJSONString():

Usage:

```
PricingUnitLabel$toJSONString()
```

Method fromJSONString():

Usage:

```
PricingUnitLabel$fromJSONString(PricingUnitLabelJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
PricingUnitLabel$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

PublicShareFilter *PublicShareFilter*

Description

PublicShareFilter Class

Format

An R6Class generator object

Methods

Public methods:

- PublicShareFilter$new()
- PublicShareFilter$toJSON()
- PublicShareFilter$fromJSON()
- PublicShareFilter$toJSONString()
- PublicShareFilter$fromJSONString()
- PublicShareFilter$clone()

Method `new()`**:**

Usage:

```
PublicShareFilter$new(...)
```

Method `toJSON()`**:**

Usage:

```
PublicShareFilter$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
PublicShareFilter$fromJSON(PublicShareFilterJson)
```

Method `toJSONString()`**:**

Usage:

```
PublicShareFilter$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
PublicShareFilter$fromJSONString(PublicShareFilterJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PublicShareFilter$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Query` *Query*

Description

Query Class

Format

An `R6Class` generator object

Public fields

`type` Querytype

`layout` Layout

`status` Querystatus

`attributeBufferHeaders` list(AttributeBufferHeader)

`writer` Writer [optional]

`reader` QueryReader [optional]

`array` Array

`totalFixedLengthBufferBytes` integer

`totalVarLenBufferBytes` integer

Methods

Public methods:

- `Query$new()`
- `Query$toJSON()`
- `Query$fromJSON()`
- `Query$toJSONString()`
- `Query$fromJSONString()`
- `Query$clone()`

Method `new()`:

Usage:

```
Query$new(
  type,
  layout,
  status,
  attributeBufferHeaders,
  array,
  totalFixedLengthBufferBytes,
  totalVarLenBufferBytes,
```

```
  writer = NULL,
  reader = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
Query$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
Query$fromJSON(QueryJson)
```

Method `toJSONString()`**:**

Usage:

```
Query$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Query$fromJSONString(QueryJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Query$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`QueryApi` *Query operations*

Description

tiledbcloud.Query

Format

An `R6Class` generator object

Methods

FinalizeQuery send a query to run against a specified array/URI registered to a group/project

@*param* namespace character

- @*param* array character
- @*param* type character
- @*param* content.type character
- @*param* query Query
- @*param* x.payer character
- @*param* open.at integer
- @*returnType* Query

- status code : 200 | query completed and results are returned in query object
- return type : Query
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

- status code : 204 | query completed successfully with no return
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetEstResultSizes send a query to run against a specified array/URI registered to a group/project

@*param* namespace character

- @*param* array character
- @*param* type character
- @*param* content.type character
- @*param* query Query
- @*param* x.payer character
- @*param* open.at integer
- @*returnType* Query

- status code : 200 | query est result size computed and results are returned in query object
- return type : Query
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

- status code : 204 | query completed successfully with no return

- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetFile send a query to run against a specified array/URI registered to a group/project, returns file bytes

@*param* namespace character
- @*param* array character
- @*param* content.type character
- @*param* x.payer character
- status code : 200 | query completed and result bytes are returned
- return type : data.frame
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

- status code : 0 | error response
- return type : Error
- response headers :

SubmitQuery send a query to run against a specified array/URI registered to a group/project

@*param* namespace character
- @*param* array character
- @*param* type character
- @*param* content.type character
- @*param* query Query
- @*param* x.payer character
- @*param* open.at integer
- @*returnType* Query

- status code : 200 | query completed and results are returned in query object
- return type : Query
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

- status code : 204 | query completed successfully with no return

- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

SubmitQueryJson send a query to run against a specified array/URI registered to a group/project, returns JSON results

@*param* namespace character

- @*param* array character
- @*param* content.type character
- @*param* query.json QueryJson
- @*param* x.payer character
- status code : 200 | query completed and results are returned in JSON format
- return type : object
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `QueryApi$new()`
- `QueryApi$FinalizeQuery()`
- `QueryApi$FinalizeQueryWithHttpInfo()`
- `QueryApi$GetEstResultSizes()`
- `QueryApi$GetEstResultSizesWithHttpInfo()`
- `QueryApi$GetFile()`
- `QueryApi$GetFileWithHttpInfo()`
- `QueryApi$SubmitQuery()`
- `QueryApi$SubmitQueryWithHttpInfo()`
- `QueryApi$SubmitQueryJson()`
- `QueryApi$SubmitQueryJsonWithHttpInfo()`

- QueryApi$clone()

Method `new()`**:**

Usage:

```
QueryApi$new(apiClient)
```

Method `FinalizeQuery()`**:**

Usage:

```
QueryApi$FinalizeQuery(
  namespace,
  array,
  type,
  content.type,
  query,
  x.payer = NULL,
  open.at = NULL,
  ...
)
```

Method `FinalizeQueryWithHttpInfo()`**:**

Usage:

```
QueryApi$FinalizeQueryWithHttpInfo(
  namespace,
  array,
  type,
  content.type,
  query,
  x.payer = NULL,
  open.at = NULL,
  ...
)
```

Method `GetEstResultSizes()`**:**

Usage:

```
QueryApi$GetEstResultSizes(
  namespace,
  array,
  type,
  content.type,
  query,
  x.payer = NULL,
  open.at = NULL,
  ...
)
```

Method `GetEstResultSizesWithHttpInfo()`**:**

Usage:

```
QueryApi$GetEstResultSizesWithHttpInfo(
  namespace,
  array,
  type,
  content.type,
  query,
  x.payer = NULL,
  open.at = NULL,
  ...
)
```

Method `GetFile()`**:**

Usage:

```
QueryApi$GetFile(namespace, array, content.type, x.payer = NULL, ...)
```

Method `GetFileWithHttpInfo()`**:**

Usage:

```
QueryApi$GetFileWithHttpInfo(
  namespace,
  array,
  content.type,
  x.payer = NULL,
  ...
)
```

Method `SubmitQuery()`**:**

Usage:

```
QueryApi$SubmitQuery(
  namespace,
  array,
  type,
  content.type,
  query,
  x.payer = NULL,
  open.at = NULL,
  ...
)
```

Method `SubmitQueryWithHttpInfo()`**:**

Usage:

```
QueryApi$SubmitQueryWithHttpInfo(
  namespace,
  array,
  type,
  content.type,
  query,
  x.payer = NULL,
  open.at = NULL,
```

```
  ...
)
```

Method `SubmitQueryJson()`**:**

Usage:

```
QueryApi$SubmitQueryJson(
  namespace,
  array,
  content.type,
  query.json,
  x.payer = NULL,
  ...
)
```

Method `SubmitQueryJsonWithHttpInfo()`**:**

Usage:

```
QueryApi$SubmitQueryJsonWithHttpInfo(
  namespace,
  array,
  content.type,
  query.json,
  x.payer = NULL,
  ...
)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
QueryApi$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## Not run:
####################  FinalizeQuery  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.type <- 'type_example' # character | type of query
var.content.type <- 'application/json' # character | Content Type of input and return mime
var.query <- Query$new() # Query | query to run
var.x.payer <- 'x.payer_example' # character | Name of organization or user who should be charged for this request
var.open.at <- 56 # integer | open_at for array in unix epoch

api.instance <- QueryApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';
```

```
#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$FinalizeQuery(var.namespace, var.array, var.type, var.content.type, var.query, x.payer=var


####################  GetEstResultSizes  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.type <- 'type_example' # character | type of query
var.content.type <- 'application/json' # character | Content Type of input and return mime
var.query <- Query$new() # Query | query to run
var.x.payer <- 'x.payer_example' # character | Name of organization or user who should be charged for this request
var.open.at <- 56 # integer | open_at for array in unix epoch

api.instance <- QueryApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetEstResultSizes(var.namespace, var.array, var.type, var.content.type, var.query, x.payer


####################  GetFile  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.content.type <- 'application/json' # character | Content Type of input and return mime
var.x.payer <- 'x.payer_example' # character | Name of organization or user who should be charged for this request

api.instance <- QueryApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';
```

```
result <- api.instance$GetFile(var.namespace, var.array, var.content.type, x.payer=var.x.payer)


####################  SubmitQuery  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.type <- 'type_example' # character | type of query
var.content.type <- 'application/json' # character | Content Type of input and return mime
var.query <- Query$new() # Query | query to run
var.x.payer <- 'x.payer_example' # character | Name of organization or user who should be charged for this request
var.open.at <- 56 # integer | open_at for array in unix epoch

api.instance <- QueryApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$SubmitQuery(var.namespace, var.array, var.type, var.content.type, var.query, x.payer=var.x


####################  SubmitQueryJson  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.content.type <- 'application/json' # character | Content Type of input and return mime
var.query.json <- QueryJson$new() # QueryJson | query to run
var.x.payer <- 'x.payer_example' # character | Name of organization or user who should be charged for this request

api.instance <- QueryApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$SubmitQueryJson(var.namespace, var.array, var.content.type, var.query.json, x.payer=var.x
```

```
## End(Not run)
```

QueryJson *QueryJson*

Description

QueryJson Class

Format

An R6Class generator object

Public fields

`query_ranges` QueryRanges [optional]

`fields` list(character) [optional]

Methods

Public methods:

- QueryJson$new()
- QueryJson$toJSON()
- QueryJson$fromJSON()
- QueryJson$toJSONString()
- QueryJson$fromJSONString()
- QueryJson$clone()

Method `new()`:

Usage:

```
QueryJson$new(query_ranges = NULL, fields = NULL, ...)
```

Method `toJSON()`:

Usage:

```
QueryJson$toJSON()
```

Method `fromJSON()`:

Usage:

```
QueryJson$fromJSON(QueryJsonJson)
```

Method `toJSONString()`:

Usage:

```
QueryJson$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
QueryJson$fromJSONString(QueryJsonJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
QueryJson$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`QueryRanges` *QueryRanges*

Description

QueryRanges Class

Format

An `R6Class` generator object

Public fields

`layout` Layout [optional]

`ranges` list(array[numeric]) [optional]

Methods

Public methods:

- `QueryRanges$new()`
- `QueryRanges$toJSON()`
- `QueryRanges$fromJSON()`
- `QueryRanges$toJSONString()`
- `QueryRanges$fromJSONString()`
- `QueryRanges$clone()`

Method `new()`**:**

Usage:

```
QueryRanges$new(layout = NULL, ranges = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
QueryRanges$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
QueryRanges$fromJSON(QueryRangesJson)
```

Method `toJSONString()`**:**

Usage:

```
QueryRanges$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
QueryRanges$fromJSONString(QueryRangesJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
QueryRanges$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`QueryReader` *QueryReader*

Description

QueryReader Class

Format

An `R6Class` generator object

Public fields

`layout` Layout [optional]

`subarray` Subarray [optional]

`readState` ReadState [optional]

`varOffsetsMode` character [optional]

`varOffsetsAddExtraElement` character [optional]

`varOffsetsBitsize` integer [optional]

Methods

Public methods:

- QueryReader$new()
- QueryReader$toJSON()
- QueryReader$fromJSON()
- QueryReader$toJSONString()
- QueryReader$fromJSONString()
- QueryReader$clone()

Method new():

Usage:

```
QueryReader$new(
  layout = NULL,
  subarray = NULL,
  readState = NULL,
  varOffsetsMode = NULL,
  varOffsetsAddExtraElement = NULL,
  varOffsetsBitsize = NULL,
  ...
)
```

Method toJSON():

Usage:

```
QueryReader$toJSON()
```

Method fromJSON():

Usage:

```
QueryReader$fromJSON(QueryReaderJson)
```

Method toJSONString():

Usage:

```
QueryReader$toJSONString()
```

Method fromJSONString():

Usage:

```
QueryReader$fromJSONString(QueryReaderJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
QueryReader$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Querystatus *Querystatus*

Description

Querystatus Class

Format

An R6Class generator object

Methods

Public methods:

- Querystatus$new()
- Querystatus$toJSON()
- Querystatus$fromJSON()
- Querystatus$toJSONString()
- Querystatus$fromJSONString()
- Querystatus$clone()

Method new():

Usage:

```
Querystatus$new(...)
```

Method toJSON():

Usage:

```
Querystatus$toJSON()
```

Method fromJSON():

Usage:

```
Querystatus$fromJSON(QuerystatusJson)
```

Method toJSONString():

Usage:

```
Querystatus$toJSONString()
```

Method fromJSONString():

Usage:

```
Querystatus$fromJSONString(QuerystatusJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
Querystatus$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Querytype *Querytype*

Description

Querytype Class

Format

An R6Class generator object

Methods

Public methods:

- Querytype$new()
- Querytype$toJSON()
- Querytype$fromJSON()
- Querytype$toJSONString()
- Querytype$fromJSONString()
- Querytype$clone()

Method new()**:**

Usage:

```
Querytype$new(...)
```

Method toJSON()**:**

Usage:

```
Querytype$toJSON()
```

Method fromJSON()**:**

Usage:

```
Querytype$fromJSON(QuerytypeJson)
```

Method toJSONString()**:**

Usage:

```
Querytype$toJSONString()
```

Method fromJSONString()**:**

Usage:

```
Querytype$fromJSONString(QuerytypeJson)
```

Method clone()**:** The objects of this class are cloneable with this method.

Usage:

```
Querytype$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

ReadState *ReadState*

Description

ReadState Class

Format

An R6Class generator object

Public fields

`initialized` character [optional]

`overflowed` character [optional]

`unsplittable` character [optional]

`subarrayPartitioner` SubarrayPartitioner [optional]

Methods

Public methods:

- `ReadState$new()`
- `ReadState$toJSON()`
- `ReadState$fromJSON()`
- `ReadState$toJSONString()`
- `ReadState$fromJSONString()`
- `ReadState$clone()`

Method `new()`:

Usage:

```
ReadState$new(
  initialized = NULL,
  overflowed = NULL,
  unsplittable = NULL,
  subarrayPartitioner = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
ReadState$toJSON()
```

Method `fromJSON()`:

Usage:

```
ReadState$fromJSON(ReadStateJson)
```

Method `toJSONString()`**:**

Usage:

```
ReadState$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ReadState$fromJSONString(ReadStateJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ReadState$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`register_array` *Register an existing array on TileDB Cloud*

Description

The underlying storage must already exist.

Usage

```
register_array(
  namespace = NULL,
  array_name,
  uri,
  description = NULL,
  access_credentials_name = NULL
)
```

Arguments

`namespace`	Namespace within TileDB cloud to charge. If this is null, the logged-in user's username will be used for the namespace.
`array_name`	The name to call the array in TileDB Cloud.
`uri`	The URI of where the array is stored.
`description`	Optional description field for the array.
`access_credentials_name`	Credentials to access the array storage. If omitted, the logged-in user's default credentials will be used.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_udf(), execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_udf(), update_udf_info(), user_profile()

`register_udf` *Register a UDF on TileDB Cloud*

Description

Registers a user-defined function on TileDB Cloud, so that it may be invoked by name later.

Usage

```
register_udf(
  namespace = NULL,
  name,
  type,
  func,
  func_text = NULL,
  version = NULL,
  image_name = NULL,
  readme = NULL,
  license_id = NULL,
  license_text = NULL,
  tags = NULL
)
```

Arguments

namespace	Namespace for the UDF to be stored in, e.g. mynamespace. If omitted, defaults to username.
name	character Name for the function to be stored under in TileDB Cloud, e.g. myudfname.
type	character One of generic, single_array, or multi_array.
func	An R function which takes a dataframe as first argument.
version	character Optional version string.
image_name	character
readme	README text to be displayed in the TileDB Cloud UI.
license_id	character See the TileDB Cloud UI for options.
license_text	character See the TileDB Cloud UI for options.
tags	list(character) Tags to apply to the UDF.
exec_raw	character Text to display in the TileDB Cloud UI's Preview tab. If omitted, a full deparse of func is used. You can set this to something shorter for brevity if you like.

Value

No return value.

See Also

Other manual-layer functions: `array_info()`, `compute_sequentially()`, `compute()`, `delayed_args<-()`, `delayed_args()`, `delayed_array_udf()`, `delayed_generic_udf()`, `delayed_sql()`, `delayed()`, `deregister_array()`, `deregister_group()`, `deregister_udf()`, `execute_array_udf()`, `execute_generic_udf()`, `execute_multi_array_udf()`, `execute_sql_query()`, `get_udf_info()`, `group_info()`, `list_arrays()`, `list_groups()`, `login()`, `register_array()`, `update_udf_info()`, `user_profile()`

`ResultFormat` *ResultFormat*

Description

ResultFormat Class

Format

An `R6Class` generator object

Methods

Public methods:

- `ResultFormat$new()`
- `ResultFormat$toJSON()`
- `ResultFormat$fromJSON()`
- `ResultFormat$toJSONString()`
- `ResultFormat$fromJSONString()`
- `ResultFormat$clone()`

Method `new()`:

Usage:

```
ResultFormat$new(...)
```

Method `toJSON()`:

Usage:

```
ResultFormat$toJSON()
```

Method `fromJSON()`:

Usage:

```
ResultFormat$fromJSON(ResultFormatJson)
```

Method `toJSONString()`:

Usage:

```
ResultFormat$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
ResultFormat$fromJSONString(ResultFormatJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ResultFormat$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`SqlApi` *Sql operations*

Description

tiledbcloud.Sql

Format

An `R6Class` generator object

Methods

RunSQL Run a sql query

@*param* namespace character

- @*param* sql SQLParameters
- @*param* accept.encoding character
- status code : 200 | JSON results in array of objects form, if the query returns results
- return type : array[object]
- response headers :

  X-TILEDB-CLOUD-TASK-ID Task ID for just completed request

- status code : 204 | SQL executed successfully
- response headers :

  X-TILEDB-CLOUD-TASK-ID Task ID for just completed request

- status code : 0 | error response
- return type : Error

- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `SqlApi$new()`
- `SqlApi$RunSQL()`
- `SqlApi$RunSQLWithHttpInfo()`
- `SqlApi$clone()`

Method `new()`:

Usage:

```
SqlApi$new(apiClient)
```

Method `RunSQL()`:

Usage:

```
SqlApi$RunSQL(namespace, sql, accept.encoding = NULL, ...)
```

Method `RunSQLWithHttpInfo()`:

Usage:

```
SqlApi$RunSQLWithHttpInfo(namespace, sql, accept.encoding = NULL, ...)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
SqlApi$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
####################  RunSQL  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace to run task under is in (an organization name or user's
var.sql <- SQLParameters$new() # SQLParameters | sql being submitted
var.accept.encoding <- 'accept.encoding_example' # character | Encoding to use

api.instance <- SqlApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';
```

```
#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$RunSQL(var.namespace, var.sql, accept.encoding=var.accept.encoding)



## End(Not run)
```

SQLParameters *SQLParameters*

Description

SQLParameters Class

Format

An R6Class generator object

Public fields

`name` character [optional]

`query` character [optional]

`output_uri` character [optional]

`store_results` character [optional]

`dont_download_results` character [optional]

`resource_class` character [optional]

`result_format` ResultFormat [optional]

`init_commands` list(character) [optional]

`parameters` list(object) [optional]

`task_graph_uuid` character [optional]

`client_node_uuid` character [optional]

Methods

Public methods:

- SQLParameters$new()
- SQLParameters$toJSON()
- SQLParameters$fromJSON()
- SQLParameters$toJSONString()

- SQLParameters$fromJSONString()
- SQLParameters$clone()

Method `new()`**:**

Usage:

```
SQLParameters$new(
  name = NULL,
  query = NULL,
  output_uri = NULL,
  store_results = NULL,
  dont_download_results = NULL,
  resource_class = NULL,
  result_format = NULL,
  init_commands = NULL,
  parameters = NULL,
  task_graph_uuid = NULL,
  client_node_uuid = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
SQLParameters$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
SQLParameters$fromJSON(SQLParametersJson)
```

Method `toJSONString()`**:**

Usage:

```
SQLParameters$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
SQLParameters$fromJSONString(SQLParametersJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SQLParameters$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

SSOProvider *SSOProvider*

Description

SSOProvider Class

Format

An R6Class generator object

Methods

Public methods:

- SSOProvider$new()
- SSOProvider$toJSON()
- SSOProvider$fromJSON()
- SSOProvider$toJSONString()
- SSOProvider$fromJSONString()
- SSOProvider$clone()

Method `new()`**:**

Usage:

```
SSOProvider$new(...)
```

Method `toJSON()`**:**

Usage:

```
SSOProvider$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
SSOProvider$fromJSON(SSOProviderJson)
```

Method `toJSONString()`**:**

Usage:

```
SSOProvider$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
SSOProvider$fromJSONString(SSOProviderJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SSOProvider$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`StatsApi` *Stats operations*

Description

tiledbcloud.Stats

Format

An `R6Class` generator object

Methods

GetTiledbStats Fetch libtiledb stat

@returnType InlineResponse200

- status code : 200 | stats
- return type : InlineResponse200
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `StatsApi$new()`
- `StatsApi$GetTiledbStats()`
- `StatsApi$GetTiledbStatsWithHttpInfo()`
- `StatsApi$clone()`

Method `new()`:

Usage:

```
StatsApi$new(apiClient)
```

Method `GetTiledbStats()`:

Usage:

```
StatsApi$GetTiledbStats(...)
```

Method `GetTiledbStatsWithHttpInfo()`**:**

Usage:

```
StatsApi$GetTiledbStatsWithHttpInfo(...)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
StatsApi$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
####################  GetTiledbStats  ####################

library(tiledbcloud)

api.instance <- StatsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetTiledbStats()



## End(Not run)
```

`Subarray` *Subarray*

Description

Subarray Class

Format

An `R6Class` generator object

Public fields

`layout` Layout [optional]

`ranges` list(SubarrayRanges) [optional]

Methods

Public methods:

- Subarray$new()
- Subarray$toJSON()
- Subarray$fromJSON()
- Subarray$toJSONString()
- Subarray$fromJSONString()
- Subarray$clone()

Method `new()`**:**

Usage:

```
Subarray$new(layout = NULL, ranges = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
Subarray$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
Subarray$fromJSON(SubarrayJson)
```

Method `toJSONString()`**:**

Usage:

```
Subarray$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Subarray$fromJSONString(SubarrayJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Subarray$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

SubarrayPartitioner *SubarrayPartitioner*

Description

SubarrayPartitioner Class

Format

An R6Class generator object

Public fields

`subarray` Subarray [optional]

`budget` list(AttributeBufferSize) [optional]

`current` SubarrayPartitionerCurrent [optional]

`state` SubarrayPartitionerState [optional]

`memoryBudget` integer [optional]

`memoryBudgetVar` integer [optional]

Methods

Public methods:

- `SubarrayPartitioner$new()`
- `SubarrayPartitioner$toJSON()`
- `SubarrayPartitioner$fromJSON()`
- `SubarrayPartitioner$toJSONString()`
- `SubarrayPartitioner$fromJSONString()`
- `SubarrayPartitioner$clone()`

Method `new()`:

Usage:

```
SubarrayPartitioner$new(
  subarray = NULL,
  budget = NULL,
  current = NULL,
  state = NULL,
  memoryBudget = NULL,
  memoryBudgetVar = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
SubarrayPartitioner$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
SubarrayPartitioner$fromJSON(SubarrayPartitionerJson)
```

Method `toJSONString()`**:**

Usage:

```
SubarrayPartitioner$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
SubarrayPartitioner$fromJSONString(SubarrayPartitionerJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SubarrayPartitioner$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`SubarrayPartitionerCurrent` *SubarrayPartitionerCurrent*

Description

SubarrayPartitionerCurrent Class

Format

An `R6Class` generator object

Public fields

`subarray` Subarray [optional]

`start` integer [optional]

`end` integer [optional]

`splitMultiRange` character [optional]

Methods

Public methods:

- SubarrayPartitionerCurrent$new()
- SubarrayPartitionerCurrent$toJSON()
- SubarrayPartitionerCurrent$fromJSON()
- SubarrayPartitionerCurrent$toJSONString()
- SubarrayPartitionerCurrent$fromJSONString()
- SubarrayPartitionerCurrent$clone()

Method `new()`:

Usage:

```
SubarrayPartitionerCurrent$new(
  subarray = NULL,
  start = NULL,
  end = NULL,
  splitMultiRange = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
SubarrayPartitionerCurrent$toJSON()
```

Method `fromJSON()`:

Usage:

```
SubarrayPartitionerCurrent$fromJSON(SubarrayPartitionerCurrentJson)
```

Method `toJSONString()`:

Usage:

```
SubarrayPartitionerCurrent$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
SubarrayPartitionerCurrent$fromJSONString(SubarrayPartitionerCurrentJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
SubarrayPartitionerCurrent$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

SubarrayPartitionerState *SubarrayPartitionerState*

Description

SubarrayPartitionerState Class

Format

An `R6Class` generator object

Public fields

`start` integer [optional]

`end` integer [optional]

`singleRange` list(Subarray) [optional]

`multiRange` list(Subarray) [optional]

Methods

Public methods:

- `SubarrayPartitionerState$new()`
- `SubarrayPartitionerState$toJSON()`
- `SubarrayPartitionerState$fromJSON()`
- `SubarrayPartitionerState$toJSONString()`
- `SubarrayPartitionerState$fromJSONString()`
- `SubarrayPartitionerState$clone()`

Method `new()`:

Usage:

```
SubarrayPartitionerState$new(
  start = NULL,
  end = NULL,
  singleRange = NULL,
  multiRange = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
SubarrayPartitionerState$toJSON()
```

Method `fromJSON()`:

Usage:

```
SubarrayPartitionerState$fromJSON(SubarrayPartitionerStateJson)
```

Method `toJSONString()`**:**

Usage:

```
SubarrayPartitionerState$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
SubarrayPartitionerState$fromJSONString(SubarrayPartitionerStateJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SubarrayPartitionerState$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`SubarrayRanges` *SubarrayRanges*

Description

SubarrayRanges Class

Format

An `R6Class` generator object

Public fields

`type` Datatype [optional]

`hasDefaultRange` character [optional]

`buffer` list(integer) [optional]

Methods

Public methods:

- `SubarrayRanges$new()`
- `SubarrayRanges$toJSON()`
- `SubarrayRanges$fromJSON()`
- `SubarrayRanges$toJSONString()`
- `SubarrayRanges$fromJSONString()`
- `SubarrayRanges$clone()`

Method `new()`**:**

Usage:

```
SubarrayRanges$new(type = NULL, hasDefaultRange = NULL, buffer = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
SubarrayRanges$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
SubarrayRanges$fromJSON(SubarrayRangesJson)
```

Method `toJSONString()`**:**

Usage:

```
SubarrayRanges$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
SubarrayRanges$fromJSONString(SubarrayRangesJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SubarrayRanges$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Subscription` *Subscription*

Description

Subscription Class

Format

An `R6Class` generator object

Public fields

`id` character [optional]

`owner_namespace_uuid` character [optional]

`customer_namespace_uuid` character [optional]

`pricing` list(Pricing) [optional]

Methods

Public methods:

- Subscription$new()
- Subscription$toJSON()
- Subscription$fromJSON()
- Subscription$toJSONString()
- Subscription$fromJSONString()
- Subscription$clone()

Method `new()`:

Usage:

```
Subscription$new(
  id = NULL,
  owner_namespace_uuid = NULL,
  customer_namespace_uuid = NULL,
  pricing = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
Subscription$toJSON()
```

Method `fromJSON()`:

Usage:

```
Subscription$fromJSON(SubscriptionJson)
```

Method `toJSONString()`:

Usage:

```
Subscription$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
Subscription$fromJSONString(SubscriptionJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
Subscription$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

TaskGraphLog *TaskGraphLog*

Description

TaskGraphLog Class

Format

An `R6Class` generator object

Public fields

`uuid` character [optional]

`namespace` character [optional]

`created_by` character [optional]

`name` character [optional]

`created_at` character [optional]

`start_time` character [optional]

`end_time` character [optional]

`status` TaskGraphLogStatus [optional]

`nodes` list(TaskGraphNodeMetadata) [optional]

Methods

Public methods:

- `TaskGraphLog$new()`
- `TaskGraphLog$toJSON()`
- `TaskGraphLog$fromJSON()`
- `TaskGraphLog$toJSONString()`
- `TaskGraphLog$fromJSONString()`
- `TaskGraphLog$clone()`

Method `new()`:

Usage:

```
TaskGraphLog$new(
  uuid = NULL,
  namespace = NULL,
  created_by = NULL,
  name = NULL,
  created_at = NULL,
  start_time = NULL,
  end_time = NULL,
```

```
  status = NULL,
  nodes = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
TaskGraphLog$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
TaskGraphLog$fromJSON(TaskGraphLogJson)
```

Method `toJSONString()`**:**

Usage:

```
TaskGraphLog$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
TaskGraphLog$fromJSONString(TaskGraphLogJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TaskGraphLog$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`TaskGraphLogsApi` *TaskGraphLogs operations*

Description

tiledbcloud.TaskGraphLogs

Format

An `R6Class` generator object

Methods

CreateTaskGraphLog Create a task graph log.

@*param* namespace character

- @*param* log TaskGraphLog
- @*returnType* TaskGraphLog

- status code : 201 | The task graph was created. The returned TaskGraphLog will include the data the client sent, with the server-defined fields added in.
- return type : TaskGraphLog
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetTaskGraphLog Fetch information about a single task graph execution.

@*param* namespace character

- @*param* id character
- @*returnType* TaskGraphLog

- status code : 200 | Information about the execution of a single task graph.
- return type : TaskGraphLog
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ListTaskGraphLogs Fetch the task graph logs of a namespace the user has access to. The returned entries will include only summary data, and will not include information about the individual tasks that were executed. (This information is available when requesting an individual task graph log.) Entries in the response are ordered from newest to oldest. Pagination parameters work as in other API methods; see PaginationMetadata.

@*param* namespace character

- @*param* created.by character
- @*param* search character
- @*param* start.time character
- @*param* end.time character
- @*param* page integer
- @*param* per.page integer

- *@returnType* TaskGraphLogsData

- status code : 200 | The task graph logs that matched the user's query.
- return type : TaskGraphLogsData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

UpdateTaskGraphLog Update information about a single task graph execution.

@param namespace character

- *@param* id character
- *@param* log TaskGraphLog
- status code : 204 | Log entry updated successfully.
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `TaskGraphLogsApi$new()`
- `TaskGraphLogsApi$CreateTaskGraphLog()`
- `TaskGraphLogsApi$CreateTaskGraphLogWithHttpInfo()`
- `TaskGraphLogsApi$GetTaskGraphLog()`
- `TaskGraphLogsApi$GetTaskGraphLogWithHttpInfo()`
- `TaskGraphLogsApi$ListTaskGraphLogs()`
- `TaskGraphLogsApi$ListTaskGraphLogsWithHttpInfo()`
- `TaskGraphLogsApi$UpdateTaskGraphLog()`
- `TaskGraphLogsApi$UpdateTaskGraphLogWithHttpInfo()`
- `TaskGraphLogsApi$clone()`

Method `new()`:

Usage:

```
TaskGraphLogsApi$new(apiClient)
```

Method `CreateTaskGraphLog()`**:**

Usage:

```
TaskGraphLogsApi$CreateTaskGraphLog(namespace, log, ...)
```

Method `CreateTaskGraphLogWithHttpInfo()`**:**

Usage:

```
TaskGraphLogsApi$CreateTaskGraphLogWithHttpInfo(namespace, log, ...)
```

Method `GetTaskGraphLog()`**:**

Usage:

```
TaskGraphLogsApi$GetTaskGraphLog(namespace, id, ...)
```

Method `GetTaskGraphLogWithHttpInfo()`**:**

Usage:

```
TaskGraphLogsApi$GetTaskGraphLogWithHttpInfo(namespace, id, ...)
```

Method `ListTaskGraphLogs()`**:**

Usage:

```
TaskGraphLogsApi$ListTaskGraphLogs(
  namespace = NULL,
  created.by = NULL,
  search = NULL,
  start.time = NULL,
  end.time = NULL,
  page = NULL,
  per.page = NULL,
  ...
)
```

Method `ListTaskGraphLogsWithHttpInfo()`**:**

Usage:

```
TaskGraphLogsApi$ListTaskGraphLogsWithHttpInfo(
  namespace = NULL,
  created.by = NULL,
  search = NULL,
  start.time = NULL,
  end.time = NULL,
  page = NULL,
  per.page = NULL,
  ...
)
```

Method `UpdateTaskGraphLog()`**:**

Usage:

```
TaskGraphLogsApi$UpdateTaskGraphLog(namespace, id, log, ...)
```

Method `UpdateTaskGraphLogWithHttpInfo()`**:**

Usage:

```
TaskGraphLogsApi$UpdateTaskGraphLogWithHttpInfo(namespace, id, log, ...)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TaskGraphLogsApi$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## Not run:
####################  CreateTaskGraphLog  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace that will own this task graph log.
var.log <- TaskGraphLog$new() # TaskGraphLog |

api.instance <- TaskGraphLogsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$CreateTaskGraphLog(var.namespace, var.log)


####################  GetTaskGraphLog  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace that owns this task graph log.
var.id <- 'id_example' # character | The UUID of the task graph log entry.

api.instance <- TaskGraphLogsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetTaskGraphLog(var.namespace, var.id)
```

```
####################  ListTaskGraphLogs  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | Include logs for this namespace.
var.created.by <- 'created.by_example' # character | Include logs from only this user.
var.search <- 'search_example' # character | search string that will look at name.
var.start.time <- 'start.time_example' # character | Include logs created after this time.
var.end.time <- 'end.time_example' # character | Include logs created before this time.
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit

api.instance <- TaskGraphLogsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ListTaskGraphLogs(namespace=var.namespace, created.by=var.created.by, search=var.search,


####################  UpdateTaskGraphLog  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | The namespace that owns this task graph log.
var.id <- 'id_example' # character | The UUID of the task graph log entry.
var.log <- TaskGraphLog$new() # TaskGraphLog | Updates to make to the task graph log. The only manual update that a c

api.instance <- TaskGraphLogsApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$UpdateTaskGraphLog(var.namespace, var.id, var.log)



## End(Not run)
```

TaskGraphLogsData *TaskGraphLogsData*

Description

TaskGraphLogsData Class

Format

An R6Class generator object

Public fields

`task_graph_logs` list(TaskGraphLog) [optional]

`pagination_metadata` PaginationMetadata [optional]

Methods

Public methods:

- `TaskGraphLogsData$new()`
- `TaskGraphLogsData$toJSON()`
- `TaskGraphLogsData$fromJSON()`
- `TaskGraphLogsData$toJSONString()`
- `TaskGraphLogsData$fromJSONString()`
- `TaskGraphLogsData$clone()`

Method `new()`:

Usage:

```
TaskGraphLogsData$new(task_graph_logs = NULL, pagination_metadata = NULL, ...)
```

Method `toJSON()`:

Usage:

```
TaskGraphLogsData$toJSON()
```

Method `fromJSON()`:

Usage:

```
TaskGraphLogsData$fromJSON(TaskGraphLogsDataJson)
```

Method `toJSONString()`:

Usage:

```
TaskGraphLogsData$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
TaskGraphLogsData$fromJSONString(TaskGraphLogsDataJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TaskGraphLogsData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`TaskGraphLogStatus` *TaskGraphLogStatus*

Description

TaskGraphLogStatus Class

Format

An `R6Class` generator object

Methods

Public methods:

- `TaskGraphLogStatus$new()`
- `TaskGraphLogStatus$toJSON()`
- `TaskGraphLogStatus$fromJSON()`
- `TaskGraphLogStatus$toJSONString()`
- `TaskGraphLogStatus$fromJSONString()`
- `TaskGraphLogStatus$clone()`

Method `new()`**:**

Usage:

```
TaskGraphLogStatus$new(...)
```

Method `toJSON()`**:**

Usage:

```
TaskGraphLogStatus$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
TaskGraphLogStatus$fromJSON(TaskGraphLogStatusJson)
```

Method `toJSONString()`**:**

Usage:

```
TaskGraphLogStatus$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
TaskGraphLogStatus$fromJSONString(TaskGraphLogStatusJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TaskGraphLogStatus$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`TaskGraphNodeMetadata` *TaskGraphNodeMetadata*

Description

TaskGraphNodeMetadata Class

Format

An `R6Class` generator object

Public fields

`client_node_uuid` character [optional]

`name` character [optional]

`depends_on` list(character) [optional]

`executions` list(ArrayTask) [optional]

Methods

Public methods:

- `TaskGraphNodeMetadata$new()`
- `TaskGraphNodeMetadata$toJSON()`
- `TaskGraphNodeMetadata$fromJSON()`
- `TaskGraphNodeMetadata$toJSONString()`
- `TaskGraphNodeMetadata$fromJSONString()`
- `TaskGraphNodeMetadata$clone()`

Method `new()`**:**

Usage:

```
TaskGraphNodeMetadata$new(
  client_node_uuid = NULL,
  name = NULL,
  depends_on = NULL,
  executions = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
TaskGraphNodeMetadata$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
TaskGraphNodeMetadata$fromJSON(TaskGraphNodeMetadataJson)
```

Method `toJSONString()`**:**

Usage:

```
TaskGraphNodeMetadata$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
TaskGraphNodeMetadata$fromJSONString(TaskGraphNodeMetadataJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TaskGraphNodeMetadata$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`TasksApi` *Tasks operations*

Description

tiledbcloud.Tasks

Format

An `R6Class` generator object

Methods

RunSQL Run a sql query

@*param* namespace character

- @*param* sql SQLParameters
- @*param* accept.encoding character
- status code : 200 | JSON results in array of objects form, if the query returns results
- return type : array[object]
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

- status code : 204 | SQL executed successfully
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

- status code : 0 | error response
- return type : Error
- response headers :

TaskIdGet Fetch an array task

@*param* id character

- @*returnType* ArrayTask

- status code : 200 | Array task
- return type : ArrayTask
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

TaskIdResultGet Retrieve results of an array task

@*param* id character

- @*param* accept.encoding character
- status code : 200 | output and format of originating request
- return type : character
- response headers :

Content-Type format results are delivered in

- status code : 202 | task is still executing
- response headers :

- status code : 404 | results were not saved, or results have expored
- return type : Error
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

TasksGet Fetch a list of all array tasks a user has access to

@*param* namespace character

- @*param* created.by character
- @*param* array character
- @*param* start integer
- @*param* end integer
- @*param* page integer
- @*param* per.page integer
- @*param* type character
- @*param* exclude.type list(character)
- @*param* file.type list(character)
- @*param* exclude.file.type list(character)
- @*param* status character
- @*param* search character
- @*param* orderby character
- @*returnType* ArrayTaskData

- status code : 200 | Array of all tasks user has access too
- return type : ArrayTaskData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `TasksApi$new()`
- `TasksApi$RunSQL()`
- `TasksApi$RunSQLWithHttpInfo()`
- `TasksApi$TaskIdGet()`
- `TasksApi$TaskIdGetWithHttpInfo()`
- `TasksApi$TaskIdResultGet()`
- `TasksApi$TaskIdResultGetWithHttpInfo()`
- `TasksApi$TasksGet()`
- `TasksApi$TasksGetWithHttpInfo()`
- `TasksApi$clone()`

Method `new()`:

Usage:

```
TasksApi$new(apiClient)
```

Method `RunSQL()`:

Usage:

```
TasksApi$RunSQL(namespace, sql, accept.encoding = NULL, ...)
```

Method `RunSQLWithHttpInfo()`:

Usage:

```
TasksApi$RunSQLWithHttpInfo(namespace, sql, accept.encoding = NULL, ...)
```

Method `TaskIdGet()`:

Usage:

```
TasksApi$TaskIdGet(id, ...)
```

Method `TaskIdGetWithHttpInfo()`:

Usage:

```
TasksApi$TaskIdGetWithHttpInfo(id, ...)
```

Method `TaskIdResultGet()`:

Usage:

```
TasksApi$TaskIdResultGet(id, accept.encoding = NULL, ...)
```

Method `TaskIdResultGetWithHttpInfo()`:

Usage:

```
TasksApi$TaskIdResultGetWithHttpInfo(id, accept.encoding = NULL, ...)
```

Method `TasksGet()`**:**

Usage:

```
TasksApi$TasksGet(
  namespace = NULL,
  created.by = NULL,
  array = NULL,
  start = NULL,
  end = NULL,
  page = NULL,
  per.page = NULL,
  type = NULL,
  exclude.type = NULL,
  file.type = NULL,
  exclude.file.type = NULL,
  status = NULL,
  search = NULL,
  orderby = NULL,
  ...
)
```

Method `TasksGetWithHttpInfo()`**:**

Usage:

```
TasksApi$TasksGetWithHttpInfo(
  namespace = NULL,
  created.by = NULL,
  array = NULL,
  start = NULL,
  end = NULL,
  page = NULL,
  per.page = NULL,
  type = NULL,
  exclude.type = NULL,
  file.type = NULL,
  exclude.file.type = NULL,
  status = NULL,
  search = NULL,
  orderby = NULL,
  ...
)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TasksApi$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
####################  RunSQL  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace to run task under is in (an organization name or user's
var.sql <- SQLParameters$new() # SQLParameters | sql being submitted
var.accept.encoding <- 'accept.encoding_example' # character | Encoding to use

api.instance <- TasksApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$RunSQL(var.namespace, var.sql, accept.encoding=var.accept.encoding)


####################  TaskIdGet  ####################

library(tiledbcloud)
var.id <- 'id_example' # character | task ID to fetch

api.instance <- TasksApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$TaskIdGet(var.id)


####################  TaskIdResultGet  ####################

library(tiledbcloud)
var.id <- 'id_example' # character | task ID to retrieve stored results
var.accept.encoding <- 'accept.encoding_example' # character | Encoding to use

api.instance <- TasksApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';
```

```
#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$TaskIdResultGet(var.id, accept.encoding=var.accept.encoding)


####################  TasksGet  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace to filter
var.created.by <- 'created.by_example' # character | username to filter
var.array <- 'array_example' # character | name/uri of array that is url-encoded to filter
var.start <- 56 # integer | start time for tasks to filter by
var.end <- 56 # integer | end time for tasks to filter by
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit
var.type <- 'type_example' # character | task type, \"QUERY\", \"SQL\", \"UDF\", \"GENERIC_UDF\"
var.exclude.type <- ['exclude.type_example'] # array[character] | task_type to exclude matching array in results, m
var.file.type <- ['file.type_example'] # array[character] | match file_type of task array, more than one can be incl
var.exclude.file.type <- ['exclude.file.type_example'] # array[character] | exclude file_type of task arrays, more
var.status <- 'status_example' # character | Filter to only return these statuses
var.search <- 'search_example' # character | search string that will look at name, namespace or description fields
var.orderby <- 'orderby_example' # character | sort by which field valid values include start_time, name

api.instance <- TasksApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$TasksGet(namespace=var.namespace, created.by=var.created.by, array=var.array, start=var.s



## End(Not run)
```

TileDBConfig *TileDBConfig*

Description

TileDBConfig Class

Format

An R6Class generator object

Public fields

configs named list(character) [optional]

Methods

Public methods:

- TileDBConfig$new()
- TileDBConfig$toJSON()
- TileDBConfig$fromJSON()
- TileDBConfig$toJSONString()
- TileDBConfig$fromJSONString()
- TileDBConfig$clone()

Method new():

Usage:

```
TileDBConfig$new(configs = NULL, ...)
```

Method toJSON():

Usage:

```
TileDBConfig$toJSON()
```

Method fromJSON():

Usage:

```
TileDBConfig$fromJSON(TileDBConfigJson)
```

Method toJSONString():

Usage:

```
TileDBConfig$toJSONString()
```

Method fromJSONString():

Usage:

```
TileDBConfig$fromJSONString(TileDBConfigJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
TileDBConfig$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Token *Token*

Description

Token Class

Format

An R6Class generator object

Public fields

`token` character [optional]

`name` character [optional]

`issued_at` character [optional]

`expires_at` character [optional]

`scope` character [optional]

Methods

Public methods:

- `Token$new()`
- `Token$toJSON()`
- `Token$fromJSON()`
- `Token$toJSONString()`
- `Token$fromJSONString()`
- `Token$clone()`

Method `new()`:

Usage:

```
Token$new(
  token = NULL,
  name = NULL,
  issued_at = NULL,
  expires_at = NULL,
  scope = "*",
  ...
)
```

Method `toJSON()`:

Usage:

```
Token$toJSON()
```

Method `fromJSON()`:

Usage:

```
Token$fromJSON(TokenJson)
```

Method `toJSONString()`**:**

Usage:

```
Token$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Token$fromJSONString(TokenJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Token$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`TokenRequest` *TokenRequest*

Description

TokenRequest Class

Format

An `R6Class` generator object

Public fields

`expires` character [optional]

`name` character [optional]

`scope` character [optional]

Methods

Public methods:

- `TokenRequest$new()`
- `TokenRequest$toJSON()`
- `TokenRequest$fromJSON()`
- `TokenRequest$toJSONString()`
- `TokenRequest$fromJSONString()`
- `TokenRequest$clone()`

Method `new()`**:**

Usage:

```
TokenRequest$new(expires = NULL, name = NULL, scope = "*", ...)
```

Method `toJSON()`**:**

Usage:

```
TokenRequest$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
TokenRequest$fromJSON(TokenRequestJson)
```

Method `toJSONString()`**:**

Usage:

```
TokenRequest$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
TokenRequest$fromJSONString(TokenRequestJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TokenRequest$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`TokenScope` *TokenScope*

Description

TokenScope Class

Format

An `R6Class` generator object

Methods

Public methods:

Method `new()`:

Usage:

```
TokenScope$new(...)
```

Method `toJSON()`:

Usage:

```
TokenScope$toJSON()
```

Method `fromJSON()`:

Usage:

```
TokenScope$fromJSON(TokenScopeJson)
```

Method `toJSONString()`:

Usage:

```
TokenScope$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
TokenScope$fromJSONString(TokenScopeJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
TokenScope$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

UDFActions *UDFActions*

Description

UDFActions Class

Format

An R6Class generator object

Methods

Public methods:

- UDFActions$new()
- UDFActions$toJSON()
- UDFActions$fromJSON()
- UDFActions$toJSONString()
- UDFActions$fromJSONString()
- UDFActions$clone()

Method `new()`**:**

Usage:

```
UDFActions$new(...)
```

Method `toJSON()`**:**

Usage:

```
UDFActions$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
UDFActions$fromJSON(UDFActionsJson)
```

Method `toJSONString()`**:**

Usage:

```
UDFActions$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
UDFActions$fromJSONString(UDFActionsJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UDFActions$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`UdfApi` *Udf operations*

Description

tiledbcloud.Udf

Format

An `R6Class` generator object

Methods

DeleteUDFInfo delete a registered UDF – this will remove all sharing and can not be undone

@*param* namespace character

- @*param* name character
- status code : 202 | UDF delete successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetUDFInfo get a specific UDF in the given namespace

@*param* namespace character

- @*param* name character
- @*returnType* UDFInfo

- status code : 200 | UDFInfo was retrieved successfully
- return type : UDFInfo
- response headers :

- status code : 404 | UDF not found
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetUDFInfoSharingPolicies Get all sharing details of the UDF

@*param* namespace character

- @*param* name character
- @*returnType* list(UDFSharing)

- status code : 200 | List of all specific sharing policies
- return type : array[UDFSharing]
- response headers :

- status code : 404 | UDF does not exist or user does not have permissions to view array-sharing policies
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

RegisterUDFInfo register a UDF in the given namespace

@*param* namespace character

- @*param* name character
- @*param* udf UDFInfoUpdate
- status code : 204 | UDF registered successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ShareUDFInfo Share a UDF with a user

@*param* namespace character

- @*param* name character
- @*param* udf.sharing UDFSharing
- status code : 204 | UDF shared successfully
- response headers :

- status code : 404 | UDF does not exist or user does not have permissions to share UDF
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

SubmitGenericUDF submit a generic UDF in the given namespace

@*param* namespace character

- @*param* udf GenericUDF
- @*param* accept.encoding character
- status code : 200 | UDF completed and the UDF-type specific result is returned
- return type : data.frame
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

- status code : 0 | error response
- return type : Error
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

SubmitMultiArrayUDF submit a multi-array UDF in the given namespace

@*param* namespace character

- @*param* udf MultiArrayUDF
- @*param* accept.encoding character
- status code : 200 | UDF completed and the UDF-type specific result is returned
- return type : data.frame
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

- status code : 0 | error response
- return type : Error
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

SubmitUDF send a UDF to run against a specified array/URI registered to a group/project

@*param* namespace character

- @*param* array character
- @*param* udf MultiArrayUDF
- @*param* x.payer character
- @*param* accept.encoding character
- @*param* v2 character

- status code : 200 | UDF completed and the UDF-type specific result is returned
- return type : data.frame
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

- status code : 0 | error response
- return type : Error
- response headers :

X-TILEDB-CLOUD-TASK-ID	Task ID for just completed request

UdfNamespaceArrayEndTimestampsGet retrieve a list of timestamps from the array fragment info listing in milliseconds, paginated

@*param* namespace character

- @*param* array character
- @*param* page integer
- @*param* per.page integer
- @*returnType* ArrayEndTimestampData

- status code : 200 | list of timestamps in milliseconds, paginated
- return type : ArrayEndTimestampData
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

UpdateUDFInfo update an existing registered UDF in the given namespace

@*param* namespace character

- @*param* name character
- @*param* udf UDFInfoUpdate
- status code : 204 | UDF updated successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- `UdfApi$new()`
- `UdfApi$DeleteUDFInfo()`
- `UdfApi$DeleteUDFInfoWithHttpInfo()`
- `UdfApi$GetUDFInfo()`
- `UdfApi$GetUDFInfoWithHttpInfo()`
- `UdfApi$GetUDFInfoSharingPolicies()`
- `UdfApi$GetUDFInfoSharingPoliciesWithHttpInfo()`
- `UdfApi$RegisterUDFInfo()`
- `UdfApi$RegisterUDFInfoWithHttpInfo()`
- `UdfApi$ShareUDFInfo()`
- `UdfApi$ShareUDFInfoWithHttpInfo()`
- `UdfApi$SubmitGenericUDF()`
- `UdfApi$SubmitGenericUDFWithHttpInfo()`
- `UdfApi$SubmitMultiArrayUDF()`
- `UdfApi$SubmitMultiArrayUDFWithHttpInfo()`
- `UdfApi$SubmitUDF()`
- `UdfApi$SubmitUDFWithHttpInfo()`
- `UdfApi$UdfNamespaceArrayEndTimestampsGet()`
- `UdfApi$UdfNamespaceArrayEndTimestampsGetWithHttpInfo()`
- `UdfApi$UpdateUDFInfo()`
- `UdfApi$UpdateUDFInfoWithHttpInfo()`
- `UdfApi$clone()`

Method `new()`:

Usage:

```
UdfApi$new(apiClient)
```

Method `DeleteUDFInfo()`:

Usage:

```
UdfApi$DeleteUDFInfo(namespace, name, ...)
```

Method `DeleteUDFInfoWithHttpInfo()`:

Usage:

```
UdfApi$DeleteUDFInfoWithHttpInfo(namespace, name, ...)
```

Method `GetUDFInfo()`:

Usage:

```
UdfApi$GetUDFInfo(namespace, name, ...)
```

Method `GetUDFInfoWithHttpInfo()`**:**

Usage:

```
UdfApi$GetUDFInfoWithHttpInfo(namespace, name, ...)
```

Method `GetUDFInfoSharingPolicies()`**:**

Usage:

```
UdfApi$GetUDFInfoSharingPolicies(namespace, name, ...)
```

Method `GetUDFInfoSharingPoliciesWithHttpInfo()`**:**

Usage:

```
UdfApi$GetUDFInfoSharingPoliciesWithHttpInfo(namespace, name, ...)
```

Method `RegisterUDFInfo()`**:**

Usage:

```
UdfApi$RegisterUDFInfo(namespace, name, udf, ...)
```

Method `RegisterUDFInfoWithHttpInfo()`**:**

Usage:

```
UdfApi$RegisterUDFInfoWithHttpInfo(namespace, name, udf, ...)
```

Method `ShareUDFInfo()`**:**

Usage:

```
UdfApi$ShareUDFInfo(namespace, name, udf.sharing, ...)
```

Method `ShareUDFInfoWithHttpInfo()`**:**

Usage:

```
UdfApi$ShareUDFInfoWithHttpInfo(namespace, name, udf.sharing, ...)
```

Method `SubmitGenericUDF()`**:**

Usage:

```
UdfApi$SubmitGenericUDF(namespace, udf, accept.encoding = NULL, ...)
```

Method `SubmitGenericUDFWithHttpInfo()`**:**

Usage:

```
UdfApi$SubmitGenericUDFWithHttpInfo(
  namespace,
  udf,
  accept.encoding = NULL,
  ...
)
```

Method `SubmitMultiArrayUDF()`**:**

Usage:

```
UdfApi$SubmitMultiArrayUDF(namespace, udf, accept.encoding = NULL, ...)
```

Method `SubmitMultiArrayUDFWithHttpInfo()`**:**

Usage:

```
UdfApi$SubmitMultiArrayUDFWithHttpInfo(
  namespace,
  udf,
  accept.encoding = NULL,
  ...
)
```

Method `SubmitUDF()`**:**

Usage:

```
UdfApi$SubmitUDF(
  namespace,
  array,
  udf,
  x.payer = NULL,
  accept.encoding = NULL,
  v2 = NULL,
  ...
)
```

Method `SubmitUDFWithHttpInfo()`**:**

Usage:

```
UdfApi$SubmitUDFWithHttpInfo(
  namespace,
  array,
  udf,
  x.payer = NULL,
  accept.encoding = NULL,
  v2 = NULL,
  ...
)
```

Method `UdfNamespaceArrayEndTimestampsGet()`**:**

Usage:

```
UdfApi$UdfNamespaceArrayEndTimestampsGet(
  namespace,
  array,
  page = NULL,
  per.page = NULL,
  ...
)
```

Method `UdfNamespaceArrayEndTimestampsGetWithHttpInfo()`**:**

Usage:

```
UdfApi$UdfNamespaceArrayEndTimestampsGetWithHttpInfo(
  namespace,
  array,
```

```
    page = NULL,
    per.page = NULL,
    ...
  )
```

Method `UpdateUDFInfo()`**:**

Usage:

```
UdfApi$UpdateUDFInfo(namespace, name, udf, ...)
```

Method `UpdateUDFInfoWithHttpInfo()`**:**

Usage:

```
UdfApi$UpdateUDFInfoWithHttpInfo(namespace, name, udf, ...)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UdfApi$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
####################  DeleteUDFInfo  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.name <- 'name_example' # character | name to register UDF under

api.instance <- UdfApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteUDFInfo(var.namespace, var.name)


####################  GetUDFInfo  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.name <- 'name_example' # character | name to register UDF under

api.instance <- UdfApi$new()
```

```
#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetUDFInfo(var.namespace, var.name)


####################  GetUDFInfoSharingPolicies  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.name <- 'name_example' # character | name of UDFInfo

api.instance <- UdfApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetUDFInfoSharingPolicies(var.namespace, var.name)


####################  RegisterUDFInfo  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.name <- 'name_example' # character | name to register UDF under
var.udf <- UDFInfoUpdate$new() # UDFInfoUpdate | UDF to register

api.instance <- UdfApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$RegisterUDFInfo(var.namespace, var.name, var.udf)
```

```
####################  ShareUDFInfo  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.name <- 'name_example' # character | name of UDFInfo
var.udf.sharing <- UDFSharing$new() # UDFSharing | Namespace and list of permissions to share with. An empty list of

api.instance <- UdfApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ShareUDFInfo(var.namespace, var.name, var.udf.sharing)


####################  SubmitGenericUDF  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.udf <- GenericUDF$new() # GenericUDF | UDF to run
var.accept.encoding <- 'accept.encoding_example' # character | Encoding to use

api.instance <- UdfApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$SubmitGenericUDF(var.namespace, var.udf, accept.encoding=var.accept.encoding)


####################  SubmitMultiArrayUDF  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.udf <- MultiArrayUDF$new() # MultiArrayUDF | UDF to run
var.accept.encoding <- 'accept.encoding_example' # character | Encoding to use

api.instance <- UdfApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';
```

```
#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$SubmitMultiArrayUDF(var.namespace, var.udf, accept.encoding=var.accept.encoding)


####################  SubmitUDF  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.udf <- MultiArrayUDF$new() # MultiArrayUDF | UDF to run
var.x.payer <- 'x.payer_example' # character | Name of organization or user who should be charged for this request
var.accept.encoding <- 'accept.encoding_example' # character | Encoding to use
var.v2 <- 'v2_example' # character | flag to indicate if v2 array UDFs should be used, currently in beta testing. Set

api.instance <- UdfApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$SubmitUDF(var.namespace, var.array, var.udf, x.payer=var.x.payer, accept.encoding=var.acc


####################  UdfNamespaceArrayEndTimestampsGet  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.array <- 'array_example' # character | name/uri of array that is url-encoded
var.page <- 56 # integer | pagination offset
var.per.page <- 56 # integer | pagination limit

api.instance <- UdfApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';
```

```
result <- api.instance$UdfNamespaceArrayEndTimestampsGet(var.namespace, var.array, page=var.page, per.page=var.p


####################  UpdateUDFInfo  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace array is in (an organization name or user's username)
var.name <- 'name_example' # character | name to register UDF under
var.udf <- UDFInfoUpdate$new() # UDFInfoUpdate | UDF to update

api.instance <- UdfApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$UpdateUDFInfo(var.namespace, var.name, var.udf)



## End(Not run)
```

`UDFArrayDetails` *UDFArrayDetails*

Description

UDFArrayDetails Class

Format

An `R6Class` generator object

Public fields

`uri` character [optional]

`ranges` QueryRanges [optional]

`buffers` list(character) [optional]

Methods

Public methods:

- `UDFArrayDetails$new()`

- UDFArrayDetails$toJSON()
- UDFArrayDetails$fromJSON()
- UDFArrayDetails$toJSONString()
- UDFArrayDetails$fromJSONString()
- UDFArrayDetails$clone()

Method `new()`**:**

Usage:

```
UDFArrayDetails$new(uri = NULL, ranges = NULL, buffers = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
UDFArrayDetails$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
UDFArrayDetails$fromJSON(UDFArrayDetailsJson)
```

Method `toJSONString()`**:**

Usage:

```
UDFArrayDetails$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
UDFArrayDetails$fromJSONString(UDFArrayDetailsJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UDFArrayDetails$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`UDFFavorite` *UDFFavorite*

Description

UDFFavorite Class

Format

An `R6Class` generator object

Public fields

`udf_uuid` character [optional]

`namespace` character [optional]

`name` character [optional]

Methods

Public methods:

- `UDFFavorite$new()`
- `UDFFavorite$toJSON()`
- `UDFFavorite$fromJSON()`
- `UDFFavorite$toJSONString()`
- `UDFFavorite$fromJSONString()`
- `UDFFavorite$clone()`

Method `new()`**:**

Usage:

```
UDFFavorite$new(udf_uuid = NULL, namespace = NULL, name = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
UDFFavorite$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
UDFFavorite$fromJSON(UDFFavoriteJson)
```

Method `toJSONString()`**:**

Usage:

```
UDFFavorite$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
UDFFavorite$fromJSONString(UDFFavoriteJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UDFFavorite$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

UDFFavoritesData *UDFFavoritesData*

Description

UDFFavoritesData Class

Format

An R6Class generator object

Public fields

udfs list(ArrayInfo) [optional]

pagination_metadata PaginationMetadata [optional]

Methods

Public methods:

- UDFFavoritesData$new()
- UDFFavoritesData$toJSON()
- UDFFavoritesData$fromJSON()
- UDFFavoritesData$toJSONString()
- UDFFavoritesData$fromJSONString()
- UDFFavoritesData$clone()

Method new():

Usage:

```
UDFFavoritesData$new(udfs = NULL, pagination_metadata = NULL, ...)
```

Method toJSON():

Usage:

```
UDFFavoritesData$toJSON()
```

Method fromJSON():

Usage:

```
UDFFavoritesData$fromJSON(UDFFavoritesDataJson)
```

Method toJSONString():

Usage:

```
UDFFavoritesData$toJSONString()
```

Method fromJSONString():

Usage:

```
UDFFavoritesData$fromJSONString(UDFFavoritesDataJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UDFFavoritesData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`UDFImage` *UDFImage*

Description

UDFImage Class

Format

An `R6Class` generator object

Public fields

`id` character [optional]

`name` character [optional]

`language` UDFLanguage [optional]

Methods

Public methods:

- `UDFImage$new()`
- `UDFImage$toJSON()`
- `UDFImage$fromJSON()`
- `UDFImage$toJSONString()`
- `UDFImage$fromJSONString()`
- `UDFImage$clone()`

Method `new()`**:**

Usage:

```
UDFImage$new(id = NULL, name = NULL, language = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
UDFImage$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
UDFImage$fromJSON(UDFImageJson)
```

Method `toJSONString()`:

Usage:

```
UDFImage$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
UDFImage$fromJSONString(UDFImageJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
UDFImage$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`UDFImageVersion` *UDFImageVersion*

Description

UDFImageVersion Class

Format

An `R6Class` generator object

Public fields

`id` character [optional]

`name` character [optional]

`udf_image_uuid` character [optional]

`docker_image` character [optional]

`version` numeric [optional]

`default` character [optional]

`latest` character [optional]

Methods

Public methods:

- `UDFImageVersion$new()`
- `UDFImageVersion$toJSON()`
- `UDFImageVersion$fromJSON()`
- `UDFImageVersion$toJSONString()`
- `UDFImageVersion$fromJSONString()`

- UDFImageVersion$clone()

Method `new()`**:**

Usage:

```
UDFImageVersion$new(
  id = NULL,
  name = NULL,
  udf_image_uuid = NULL,
  docker_image = NULL,
  version = NULL,
  default = NULL,
  latest = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
UDFImageVersion$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
UDFImageVersion$fromJSON(UDFImageVersionJson)
```

Method `toJSONString()`**:**

Usage:

```
UDFImageVersion$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
UDFImageVersion$fromJSONString(UDFImageVersionJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UDFImageVersion$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

UDFInfo *UDFInfo*

Description

UDFInfo Class

Format

An R6Class generator object

Public fields

`id` character [optional]

`name` character [optional]

`language` UDFLanguage [optional]

`type` UDFType [optional]

`readme` character [optional]

`license_id` character [optional]

`license_text` character [optional]

`tags` list(character) [optional]

Methods

Public methods:

- `UDFInfo$new()`
- `UDFInfo$toJSON()`
- `UDFInfo$fromJSON()`
- `UDFInfo$toJSONString()`
- `UDFInfo$fromJSONString()`
- `UDFInfo$clone()`

Method `new()`:

Usage:

```
UDFInfo$new(
  id = NULL,
  name = NULL,
  language = NULL,
  type = NULL,
  readme = NULL,
  license_id = NULL,
  license_text = NULL,
  tags = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
UDFInfo$toJSON()
```

Method `fromJSON()`:

Usage:

```
UDFInfo$fromJSON(UDFInfoJson)
```

Method `toJSONString()`:

Usage:

```
UDFInfo$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
UDFInfo$fromJSONString(UDFInfoJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
UDFInfo$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`UDFInfoUpdate` *UDFInfoUpdate*

Description

UDFInfoUpdate Class

Format

An `R6Class` generator object

Public fields

`name` character [optional]

`language` UDFLanguage [optional]

`version` character [optional]

`image_name` character [optional]

`type` UDFType [optional]

`exec` character [optional]

`exec_raw` character [optional]

`readme` character [optional]

`license_id` character [optional]

`license_text` character [optional]

`tags` list(character) [optional]

Methods

Public methods:

- UDFInfoUpdate$new()
- UDFInfoUpdate$toJSON()
- UDFInfoUpdate$fromJSON()
- UDFInfoUpdate$toJSONString()
- UDFInfoUpdate$fromJSONString()
- UDFInfoUpdate$clone()

Method `new()`:

Usage:

```
UDFInfoUpdate$new(
  name = NULL,
  language = NULL,
  version = NULL,
  image_name = NULL,
  type = NULL,
  exec = NULL,
  exec_raw = NULL,
  readme = NULL,
  license_id = NULL,
  license_text = NULL,
  tags = NULL,
  ...
)
```

Method `toJSON()`:

Usage:

```
UDFInfoUpdate$toJSON()
```

Method `fromJSON()`:

Usage:

```
UDFInfoUpdate$fromJSON(UDFInfoUpdateJson)
```

Method `toJSONString()`:

Usage:

```
UDFInfoUpdate$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
UDFInfoUpdate$fromJSONString(UDFInfoUpdateJson)
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
UDFInfoUpdate$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

UDFLanguage *UDFLanguage*

Description

UDFLanguage Class

Format

An R6Class generator object

Methods

Public methods:

- UDFLanguage$new()
- UDFLanguage$toJSON()
- UDFLanguage$fromJSON()
- UDFLanguage$toJSONString()
- UDFLanguage$fromJSONString()
- UDFLanguage$clone()

Method `new()`**:**

Usage:

```
UDFLanguage$new(...)
```

Method `toJSON()`**:**

Usage:

```
UDFLanguage$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
UDFLanguage$fromJSON(UDFLanguageJson)
```

Method `toJSONString()`**:**

Usage:

```
UDFLanguage$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
UDFLanguage$fromJSONString(UDFLanguageJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UDFLanguage$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

UDFSharing *UDFSharing*

Description

UDFSharing Class

Format

An R6Class generator object

Public fields

`actions` list(UDFActions) [optional]

`namespace` character [optional]

`namespace_type` character [optional]

Methods

Public methods:

- `UDFSharing$new()`
- `UDFSharing$toJSON()`
- `UDFSharing$fromJSON()`
- `UDFSharing$toJSONString()`
- `UDFSharing$fromJSONString()`
- `UDFSharing$clone()`

Method `new()`:

Usage:

```
UDFSharing$new(actions = NULL, namespace = NULL, namespace_type = NULL, ...)
```

Method `toJSON()`:

Usage:

```
UDFSharing$toJSON()
```

Method `fromJSON()`:

Usage:

```
UDFSharing$fromJSON(UDFSharingJson)
```

Method `toJSONString()`:

Usage:

```
UDFSharing$toJSONString()
```

Method `fromJSONString()`:

Usage:

```
UDFSharing$fromJSONString(UDFSharingJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UDFSharing$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`UDFSubarray` *UDFSubarray*

Description

UDFSubarray Class

Format

An `R6Class` generator object

Public fields

`layout` Layout [optional]

`ranges` list(UDFSubarrayRange) [optional]

Methods

Public methods:

- `UDFSubarray$new()`
- `UDFSubarray$toJSON()`
- `UDFSubarray$fromJSON()`
- `UDFSubarray$toJSONString()`
- `UDFSubarray$fromJSONString()`
- `UDFSubarray$clone()`

Method `new()`**:**

Usage:

```
UDFSubarray$new(layout = NULL, ranges = NULL, ...)
```

Method `toJSON()`**:**

Usage:

```
UDFSubarray$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
UDFSubarray$fromJSON(UDFSubarrayJson)
```

Method `toJSONString()`**:**

Usage:

```
UDFSubarray$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
UDFSubarray$fromJSONString(UDFSubarrayJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UDFSubarray$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`UDFSubarrayRange` *UDFSubarrayRange*

Description

UDFSubarrayRange Class

Format

An `R6Class` generator object

Public fields

`dimension_id` integer [optional]

`range_start` DimensionCoordinate [optional]

`range_end` DimensionCoordinate [optional]

Methods

Public methods:

- `UDFSubarrayRange$new()`
- `UDFSubarrayRange$toJSON()`
- `UDFSubarrayRange$fromJSON()`
- `UDFSubarrayRange$toJSONString()`
- `UDFSubarrayRange$fromJSONString()`
- `UDFSubarrayRange$clone()`

Method `new()`**:**

Usage:

```
UDFSubarrayRange$new(
  dimension_id = NULL,
  range_start = NULL,
  range_end = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
UDFSubarrayRange$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
UDFSubarrayRange$fromJSON(UDFSubarrayRangeJson)
```

Method `toJSONString()`**:**

Usage:

```
UDFSubarrayRange$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
UDFSubarrayRange$fromJSONString(UDFSubarrayRangeJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UDFSubarrayRange$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`UDFType` *UDFType*

Description

UDFType Class

Format

An `R6Class` generator object

Methods

Public methods:

- UDFType$new()
- UDFType$toJSON()
- UDFType$fromJSON()
- UDFType$toJSONString()
- UDFType$fromJSONString()
- UDFType$clone()

Method new():

Usage:

```
UDFType$new(...)
```

Method toJSON():

Usage:

```
UDFType$toJSON()
```

Method fromJSON():

Usage:

```
UDFType$fromJSON(UDFTypeJson)
```

Method toJSONString():

Usage:

```
UDFType$toJSONString()
```

Method fromJSONString():

Usage:

```
UDFType$fromJSONString(UDFTypeJson)
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
UDFType$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

update_udf_info *Update a UDF on TileDB Cloud*

Description

Updates information for a specified user-defined function on TileDB Cloud. Please provide all parameters to be set, not only the ones that need changing.

Usage

```
update_udf_info(
  namespace,
  name,
  type,
  func = NULL,
  func_text = NULL,
  version = NULL,
  image_name = NULL,
  readme = NULL,
  license_id = NULL,
  license_text = NULL,
  tags = NULL
)
```

Arguments

namespace	Namespace for the UDF to be stored in, e.g. mynamespace. If omitted, defaults to username.
name	character Name for the function to be stored under in TileDB Cloud, e.g. myudfname.
type	character One of generic, single_array, or multi_array.
func	An R function which takes a dataframe as first argument.
version	character Optional version string.
image_name	character
readme	README text to be displayed in the TileDB Cloud UI.
license_id	character See the TileDB Cloud UI for options.
license_text	character See the TileDB Cloud UI for options.
tags	list(character) Tags to apply to the UDF.
exec_raw	character Text to display in the TileDB Cloud UI's Preview tab. If omitted, a full deparse of func is used. You can set this to something shorter for brevity if you like.

Value

No return value.

See Also

Other manual-layer functions: array_info(), compute_sequentially(), compute(), delayed_args<-(), delayed_args(), delayed_array_udf(), delayed_generic_udf(), delayed_sql(), delayed(), deregister_array(), deregister_group(), deregister_udf(), execute_array_udf(), execute_generic_udf(), execute_multi_array_udf(), execute_sql_query(), get_udf_info(), group_info(), list_arrays(), list_groups(), login(), register_array(), register_udf(), user_profile()

`User` *User*

Description

User Class

Format

An `R6Class` generator object

Public fields

`id` character [optional]

`username` character

`password` character [optional]

`name` character [optional]

`email` character [optional]

`is_valid_email` character [optional]

`stripe_connect` character [optional]

`company` character [optional]

`logo` character [optional]

`last_activity_date` character [optional]

`timezone` character [optional]

`organizations` list(OrganizationUser) [optional]

`allowed_actions` list(NamespaceActions) [optional]

`enabled_features` list(character) [optional]

`unpaid_subscription` character [optional]

`default_s3_path` character [optional]

`default_s3_path_credentials_name` character [optional]

`default_namespace_charged` character [optional]

Methods

Public methods:

- User$new()
- User$toJSON()
- User$fromJSON()
- User$toJSONString()
- User$fromJSONString()
- User$clone()

Method new():

Usage:

```
User$new(
  username,
  id = NULL,
  password = NULL,
  name = NULL,
  email = NULL,
  is_valid_email = NULL,
  stripe_connect = NULL,
  company = NULL,
  logo = NULL,
  last_activity_date = NULL,
  timezone = NULL,
  organizations = NULL,
  allowed_actions = NULL,
  enabled_features = NULL,
  unpaid_subscription = NULL,
  default_s3_path = NULL,
  default_s3_path_credentials_name = NULL,
  default_namespace_charged = NULL,
  ...
)
```

Method toJSON():

Usage:

```
User$toJSON()
```

Method fromJSON():

Usage:

```
User$fromJSON(UserJson)
```

Method toJSONString():

Usage:

```
User$toJSONString()
```

Method fromJSONString():

Usage:

```
User$fromJSONString(UserJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
User$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`UserApi` *User operations*

Description

tiledbcloud.User

Format

An `R6Class` generator object

Methods

AddAWSAccessCredentials Add aws keys

@*param* namespace character

- @*param* aws.access.credentials AWSAccessCredentials
- status code : 204 | AWS keys added successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

AddUserToOrganization add a user to an organization

@*param* organization character

- @*param* user OrganizationUser
- status code : 204 | user added to organization successfully
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

CheckAWSAccessCredentials Check if aws keys are set

@*param* namespace character

- @*returnType* list(AWSAccessCredentials)

- status code : 200 | AWS keys are set
- return type : array[AWSAccessCredentials]
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

CheckAWSAccessCredentialsByName Check if aws keys are set by name

@*param* namespace character

- @*param* name character
- @*returnType* AWSAccessCredentials

- status code : 200 | AWS keys are set
- return type : AWSAccessCredentials
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ConfirmEmail confirm user email

status code : 204 | User email confirmed successfully

- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

CreateUser create a user

@*param* user User

- status code : 204 | user created successfully

- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

DeleteAWSAccessCredentials delete a AWS Access credentials in a namespace. This will likely cause arrays to become unreachable

@*param* namespace character

- @*param* name character
- status code : 204 | AWS credentials deleted
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

DeleteUser delete a user

@*param* username character

- status code : 204 | user deleted
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

DeleteUserFromOrganization delete a user from an organization

@*param* organization character

- @*param* username character
- status code : 204 | user delete from organization successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetOrganizationUser get a user from an organization

@*param* organization character

- *@param* username character
- *@returnType* OrganizationUser

- status code : 200 | user from organization
- return type : OrganizationUser
- response headers :

- status code : 404 | User is not in organization
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetSession Get session token for user

@param remember.me character

- *@returnType* Token

- status code : 200 | Session token
- return type : Token
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetTokenScopes retrieves available token scopes for a user

@returnType list(TokenScope)

- status code : 200 | available token scopes
- return type : array[TokenScope]
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetUser get a user

@returnType User

- status code : 200 | user details
- return type : User
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

GetUserWithUsername get a user

@param username character

- *@returnType* User

- status code : 200 | user details
- return type : User
- response headers :

- status code : 404 | User does not exist
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

RequestToken Request an authorization token, optionally taken a TokenRequest object to set parameters on the token

@param token.request TokenRequest

- *@returnType* Token

- status code : 200 | token
- return type : Token
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

ResetUserPassword reset user password

@param user InlineObject

- status code : 204 | User password updated successfully

- response headers :

- status code : 404 | User not found
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

RevokeToken revoke an authorization token

@param token character

- status code : 204 | revokation successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

TokensGet Fetch a list of user tokens

@returnType list(Token)

- status code : 200 | Array of user created non-session tokens
- return type : array[Token]
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

TokensSessionGet Fetch a list of user session tokens

@returnType list(Token)

- status code : 200 | Array of user created session tokens
- return type : array[Token]
- response headers :

- status code : 0 | error response
- return type : Error

- response headers :

UpdateAWSAccessCredentials Update aws keys or associated buckets. This will update the key associations for each array in the namespace

@*param* namespace character

- @*param* name character
- @*param* aws.access.credentials AWSAccessCredentials
- status code : 204 | AWS keys updated successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

UpdateUser update a user

@*param* username character

- @*param* user User
- status code : 204 | user updated successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

UpdateUserInOrganization update a user in an organization

@*param* organization character

- @*param* username character
- @*param* user OrganizationUser
- status code : 204 | user update in organization successfully
- response headers :

- status code : 0 | error response
- return type : Error
- response headers :

Public fields

`apiClient` Handles the client-server communication.

Methods

Public methods:

- UserApi$new()
- UserApi$AddAWSAccessCredentials()
- UserApi$AddAWSAccessCredentialsWithHttpInfo()
- UserApi$AddUserToOrganization()
- UserApi$AddUserToOrganizationWithHttpInfo()
- UserApi$CheckAWSAccessCredentials()
- UserApi$CheckAWSAccessCredentialsWithHttpInfo()
- UserApi$CheckAWSAccessCredentialsByName()
- UserApi$CheckAWSAccessCredentialsByNameWithHttpInfo()
- UserApi$ConfirmEmail()
- UserApi$ConfirmEmailWithHttpInfo()
- UserApi$CreateUser()
- UserApi$CreateUserWithHttpInfo()
- UserApi$DeleteAWSAccessCredentials()
- UserApi$DeleteAWSAccessCredentialsWithHttpInfo()
- UserApi$DeleteUser()
- UserApi$DeleteUserWithHttpInfo()
- UserApi$DeleteUserFromOrganization()
- UserApi$DeleteUserFromOrganizationWithHttpInfo()
- UserApi$GetOrganizationUser()
- UserApi$GetOrganizationUserWithHttpInfo()
- UserApi$GetSession()
- UserApi$GetSessionWithHttpInfo()
- UserApi$GetTokenScopes()
- UserApi$GetTokenScopesWithHttpInfo()
- UserApi$GetUser()
- UserApi$GetUserWithHttpInfo()
- UserApi$GetUserWithUsername()
- UserApi$GetUserWithUsernameWithHttpInfo()
- UserApi$RequestToken()
- UserApi$RequestTokenWithHttpInfo()
- UserApi$ResetUserPassword()
- UserApi$ResetUserPasswordWithHttpInfo()
- UserApi$RevokeToken()
- UserApi$RevokeTokenWithHttpInfo()
- UserApi$TokensGet()
- UserApi$TokensGetWithHttpInfo()
- UserApi$TokensSessionGet()
- UserApi$TokensSessionGetWithHttpInfo()
- UserApi$UpdateAWSAccessCredentials()

- UserApi$UpdateAWSAccessCredentialsWithHttpInfo()
- UserApi$UpdateUser()
- UserApi$UpdateUserWithHttpInfo()
- UserApi$UpdateUserInOrganization()
- UserApi$UpdateUserInOrganizationWithHttpInfo()
- UserApi$clone()

Method `new()`**:**

Usage:

```
UserApi$new(apiClient)
```

Method `AddAWSAccessCredentials()`**:**

Usage:

```
UserApi$AddAWSAccessCredentials(namespace, aws.access.credentials, ...)
```

Method `AddAWSAccessCredentialsWithHttpInfo()`**:**

Usage:

```
UserApi$AddAWSAccessCredentialsWithHttpInfo(
  namespace,
  aws.access.credentials,
  ...
)
```

Method `AddUserToOrganization()`**:**

Usage:

```
UserApi$AddUserToOrganization(organization, user, ...)
```

Method `AddUserToOrganizationWithHttpInfo()`**:**

Usage:

```
UserApi$AddUserToOrganizationWithHttpInfo(organization, user, ...)
```

Method `CheckAWSAccessCredentials()`**:**

Usage:

```
UserApi$CheckAWSAccessCredentials(namespace, ...)
```

Method `CheckAWSAccessCredentialsWithHttpInfo()`**:**

Usage:

```
UserApi$CheckAWSAccessCredentialsWithHttpInfo(namespace, ...)
```

Method `CheckAWSAccessCredentialsByName()`**:**

Usage:

```
UserApi$CheckAWSAccessCredentialsByName(namespace, name, ...)
```

Method `CheckAWSAccessCredentialsByNameWithHttpInfo()`**:**

Usage:

```
UserApi$CheckAWSAccessCredentialsByNameWithHttpInfo(namespace, name, ...)
```

Method `ConfirmEmail()`**:**

Usage:

```
UserApi$ConfirmEmail(...)
```

Method `ConfirmEmailWithHttpInfo()`**:**

Usage:

```
UserApi$ConfirmEmailWithHttpInfo(...)
```

Method `CreateUser()`**:**

Usage:

```
UserApi$CreateUser(user, ...)
```

Method `CreateUserWithHttpInfo()`**:**

Usage:

```
UserApi$CreateUserWithHttpInfo(user, ...)
```

Method `DeleteAWSAccessCredentials()`**:**

Usage:

```
UserApi$DeleteAWSAccessCredentials(namespace, name, ...)
```

Method `DeleteAWSAccessCredentialsWithHttpInfo()`**:**

Usage:

```
UserApi$DeleteAWSAccessCredentialsWithHttpInfo(namespace, name, ...)
```

Method `DeleteUser()`**:**

Usage:

```
UserApi$DeleteUser(username, ...)
```

Method `DeleteUserWithHttpInfo()`**:**

Usage:

```
UserApi$DeleteUserWithHttpInfo(username, ...)
```

Method `DeleteUserFromOrganization()`**:**

Usage:

```
UserApi$DeleteUserFromOrganization(organization, username, ...)
```

Method `DeleteUserFromOrganizationWithHttpInfo()`**:**

Usage:

```
UserApi$DeleteUserFromOrganizationWithHttpInfo(organization, username, ...)
```

Method `GetOrganizationUser()`**:**

Usage:

```
UserApi$GetOrganizationUser(organization, username, ...)
```

Method `GetOrganizationUserWithHttpInfo()`**:**

Usage:

```
UserApi$GetOrganizationUserWithHttpInfo(organization, username, ...)
```

Method `GetSession()`**:**

Usage:

```
UserApi$GetSession(remember.me = NULL, ...)
```

Method `GetSessionWithHttpInfo()`**:**

Usage:

```
UserApi$GetSessionWithHttpInfo(remember.me = NULL, ...)
```

Method `GetTokenScopes()`**:**

Usage:

```
UserApi$GetTokenScopes(...)
```

Method `GetTokenScopesWithHttpInfo()`**:**

Usage:

```
UserApi$GetTokenScopesWithHttpInfo(...)
```

Method `GetUser()`**:**

Usage:

```
UserApi$GetUser(...)
```

Method `GetUserWithHttpInfo()`**:**

Usage:

```
UserApi$GetUserWithHttpInfo(...)
```

Method `GetUserWithUsername()`**:**

Usage:

```
UserApi$GetUserWithUsername(username, ...)
```

Method `GetUserWithUsernameWithHttpInfo()`**:**

Usage:

```
UserApi$GetUserWithUsernameWithHttpInfo(username, ...)
```

Method `RequestToken()`**:**

Usage:

```
UserApi$RequestToken(token.request = NULL, ...)
```

Method `RequestTokenWithHttpInfo()`**:**

Usage:

```
UserApi$RequestTokenWithHttpInfo(token.request = NULL, ...)
```

Method `ResetUserPassword()`**:**

Usage:

```
UserApi$ResetUserPassword(user, ...)
```

Method `ResetUserPasswordWithHttpInfo()`**:**

Usage:

```
UserApi$ResetUserPasswordWithHttpInfo(user, ...)
```

Method `RevokeToken()`**:**

Usage:

```
UserApi$RevokeToken(token, ...)
```

Method `RevokeTokenWithHttpInfo()`**:**

Usage:

```
UserApi$RevokeTokenWithHttpInfo(token, ...)
```

Method `TokensGet()`**:**

Usage:

```
UserApi$TokensGet(...)
```

Method `TokensGetWithHttpInfo()`**:**

Usage:

```
UserApi$TokensGetWithHttpInfo(...)
```

Method `TokensSessionGet()`**:**

Usage:

```
UserApi$TokensSessionGet(...)
```

Method `TokensSessionGetWithHttpInfo()`**:**

Usage:

```
UserApi$TokensSessionGetWithHttpInfo(...)
```

Method `UpdateAWSAccessCredentials()`**:**

Usage:

```
UserApi$UpdateAWSAccessCredentials(
  namespace,
  name,
  aws.access.credentials,
  ...
)
```

Method `UpdateAWSAccessCredentialsWithHttpInfo()`**:**

Usage:

```
UserApi$UpdateAWSAccessCredentialsWithHttpInfo(
  namespace,
  name,
  aws.access.credentials,
  ...
)
```

Method `UpdateUser()`**:**

Usage:

```
UserApi$UpdateUser(username, user, ...)
```

Method `UpdateUserWithHttpInfo()`**:**

Usage:

```
UserApi$UpdateUserWithHttpInfo(username, user, ...)
```

Method `UpdateUserInOrganization()`**:**

Usage:

```
UserApi$UpdateUserInOrganization(organization, username, user, ...)
```

Method `UpdateUserInOrganizationWithHttpInfo()`**:**

Usage:

```
UserApi$UpdateUserInOrganizationWithHttpInfo(organization, username, user, ...)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UserApi$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## Not run:
####################  AddAWSAccessCredentials  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace
var.aws.access.credentials <- AWSAccessCredentials$new() # AWSAccessCredentials | aws access credentials to store

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$AddAWSAccessCredentials(var.namespace, var.aws.access.credentials)


####################  AddUserToOrganization  ####################

library(tiledbcloud)
var.organization <- 'organization_example' # character | organization name
```

```
var.user <- OrganizationUser$new() # OrganizationUser | user to add

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$AddUserToOrganization(var.organization, var.user)


####################  CheckAWSAccessCredentials  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$CheckAWSAccessCredentials(var.namespace)


####################  CheckAWSAccessCredentialsByName  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace
var.name <- 'name_example' # character | name

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$CheckAWSAccessCredentialsByName(var.namespace, var.name)
```

```
####################  ConfirmEmail  ####################

library(tiledbcloud)

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ConfirmEmail()


####################  CreateUser  ####################

library(tiledbcloud)
var.user <- User$new() # User | user to create

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$CreateUser(var.user)


####################  DeleteAWSAccessCredentials  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace
var.name <- 'name_example' # character | name

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
```

```
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteAWSAccessCredentials(var.namespace, var.name)


####################  DeleteUser  ####################

library(tiledbcloud)
var.username <- 'username_example' # character | username or ID

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteUser(var.username)


####################  DeleteUserFromOrganization  ####################

library(tiledbcloud)
var.organization <- 'organization_example' # character | organization name
var.username <- 'username_example' # character | username to manipulate

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$DeleteUserFromOrganization(var.organization, var.username)


####################  GetOrganizationUser  ####################

library(tiledbcloud)
var.organization <- 'organization_example' # character | organization name
var.username <- 'username_example' # character | username to manipulate

api.instance <- UserApi$new()
```

```
#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetOrganizationUser(var.organization, var.username)


####################  GetSession  ####################

library(tiledbcloud)
var.remember.me <- 'remember.me_example' # character | flag to create a token with expiration of 30 days, default is

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetSession(remember.me=var.remember.me)


####################  GetTokenScopes  ####################

library(tiledbcloud)

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetTokenScopes()


####################  GetUser  ####################

library(tiledbcloud)
```

```
api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetUser()


####################  GetUserWithUsername  ####################

library(tiledbcloud)
var.username <- 'username_example' # character | username or ID

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$GetUserWithUsername(var.username)


####################  RequestToken  ####################

library(tiledbcloud)
var.token.request <- TokenRequest$new() # TokenRequest | token request object

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$RequestToken(token.request=var.token.request)


####################  ResetUserPassword  ####################
```

```
library(tiledbcloud)
var.user <- InlineObject$new() # InlineObject |

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$ResetUserPassword(var.user)


####################  RevokeToken  ####################

library(tiledbcloud)
var.token <- 'token_example' # character | token name or token itself

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$RevokeToken(var.token)


####################  TokensGet  ####################

library(tiledbcloud)

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$TokensGet()
```

```
####################  TokensSessionGet  ####################

library(tiledbcloud)

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$TokensSessionGet()


####################  UpdateAWSAccessCredentials  ####################

library(tiledbcloud)
var.namespace <- 'namespace_example' # character | namespace
var.name <- 'name_example' # character | name
var.aws.access.credentials <- AWSAccessCredentials$new() # AWSAccessCredentials | aws credentials to update

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$UpdateAWSAccessCredentials(var.namespace, var.name, var.aws.access.credentials)


####################  UpdateUser  ####################

library(tiledbcloud)
var.username <- 'username_example' # character | username or ID
var.user <- User$new() # User | user details to update

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
```

```
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$UpdateUser(var.username, var.user)


####################  UpdateUserInOrganization  ####################

library(tiledbcloud)
var.organization <- 'organization_example' # character | organization name
var.username <- 'username_example' # character | username to manipulate
var.user <- OrganizationUser$new() # OrganizationUser | user details to update

api.instance <- UserApi$new()

#Configure API key authorization: ApiKeyAuth
api.instance$apiClient$apiKeys['X-TILEDB-REST-API-KEY'] <- 'TODO_YOUR_API_KEY';

#Configure HTTP basic authorization: BasicAuth
# provide your username in the user-serial format
api.instance$apiClient$username <- '<user-serial>';
# provide your api key generated using the developer portal
api.instance$apiClient$password <- '<api_key>';

result <- api.instance$UpdateUserInOrganization(var.organization, var.username, var.user)



## End(Not run)
```

`user_profile` *Show information from user TileDB Cloud user profile*

Description

This function shows user information for the currently logged-in account on TileDB Cloud.

Usage

```
user_profile(include_logo = FALSE)
```

Arguments

`include_logo`	If set to True, include the `logo` field in the return value. By default this is omitted since it's a long base64-encoded string which takes up a lot of screen space and is likely uninteresting.

Details

Nominally you will first call `login`; if not, the results of the last login at `~/.tiledb/cloud.json` will be used.

Value

A list of user properties from the currently logged-in TileDB cloud account.

See Also

Other manual-layer functions: `array_info()`, `compute_sequentially()`, `compute()`, `delayed_args<-()`, `delayed_args()`, `delayed_array_udf()`, `delayed_generic_udf()`, `delayed_sql()`, `delayed()`, `deregister_array()`, `deregister_group()`, `deregister_udf()`, `execute_array_udf()`, `execute_generic_udf()`, `execute_multi_array_udf()`, `execute_sql_query()`, `get_udf_info()`, `group_info()`, `list_arrays()`, `list_groups()`, `login()`, `register_array()`, `register_udf()`, `update_udf_info()`

`Writer` *Writer*

Description

Writer Class

Format

An `R6Class` generator object

Public fields

`checkCoordDups` character [optional]

`checkCoordOOB` character [optional]

`dedupCoords` character [optional]

`subarray` DomainArray [optional]

Methods

Public methods:

- `Writer$new()`
- `Writer$toJSON()`
- `Writer$fromJSON()`
- `Writer$toJSONString()`
- `Writer$fromJSONString()`
- `Writer$clone()`

Method `new()`:

Usage:

```
Writer$new(
  checkCoordDups = NULL,
  checkCoordOOB = NULL,
  dedupCoords = NULL,
  subarray = NULL,
  ...
)
```

Method `toJSON()`**:**

Usage:

```
Writer$toJSON()
```

Method `fromJSON()`**:**

Usage:

```
Writer$fromJSON(WriterJson)
```

Method `toJSONString()`**:**

Usage:

```
Writer$toJSONString()
```

Method `fromJSONString()`**:**

Usage:

```
Writer$fromJSONString(WriterJson)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Writer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index