Package 'd4storagehub4R'

April 15, 2024

Version 0.4-4

Date 2024-04-15

Title Interface to 'D4Science' 'StorageHub' API

Maintainer Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Depends R (>= 3.3.0), methods

Imports R6, httr, jsonlite, XML, xml2, keyring, tools

Suggests testthat, parallel

Description Provides an interface to 'D4Science' 'StorageHub' API (<https://dev.d4science.org/>). Allows to get user profile, and perform actions over the 'StorageHub' (workspace) including creation of folders, files management (upload/update/deletion/sharing), and listing of stored resources.

License MIT + file LICENSE

URL https://github.com/eblondel/d4storagehub4R, https://www.d4science.org/, https://dev.d4science.org/

BugReports https://github.com/eblondel/d4storagehub4R/issues

LazyLoad yes

RoxygenNote 7.3.1

Encoding UTF-8

NeedsCompilation no

Author Emmanuel Blondel [aut, cre] (<https://orcid.org/0000-0002-5870-5762>)

Repository CRAN

Date/Publication 2024-04-15 08:00:02 UTC

R topics documented:

`d4storagehub4R` *Interface to 'D4Science' 'StorageHub' API*

Description

Provides an interface to 'D4Science' 'StorageHub' API (<https://dev.d4science.org/>). Allows to get user profile, and perform actions over the 'StorageHub' (workspace) including creation of folders, files management (upload/update/deletion/sharing), and listing of stored resources.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

`d4storagehub4RLogger` *d4storagehub4RLogger*

Description

d4storagehub4RLogger

d4storagehub4RLogger

Format

`R6Class` object.

Value

Object of `R6Class` for modelling a simple logger

Abstract Methods

`INFO(text)` Logger to report information. Used internally

`WARN(text)` Logger to report warnings. Used internally

`ERROR(text)` Logger to report errors. Used internally

Public fields

`verbose.info` verbose info, default is `FALSE`

`verbose.debug` verbose debug, default is `FALSE`

`loggerType` logger type

Methods

Public methods:

Method `logger()`**:** Logger function

Usage:

```
d4storagehub4RLogger$logger(type, text)
```

Arguments:

`type` type

`text` text

Method `INFO()`**:** INFO logger function

Usage:

```
d4storagehub4RLogger$INFO(text)
```

Arguments:

`text` text

Method `WARN()`**:** WARN logger function

Usage:

```
d4storagehub4RLogger$WARN(text)
```

Arguments:

`text` text

Method `ERROR()`**:** ERROR logger function

Usage:

```
d4storagehub4RLogger$ERROR(text)
```

Arguments:

`text` text

Method `new()`**:** Initializes a basic logger class

Usage:

```
d4storagehub4RLogger$new(logger = NULL)
```

Arguments:

`logger` the type of logger, either `NULL` (default), `INFO`, or `DEBUG`

Method `getClassName()`**:** Get class name

Usage:

```
d4storagehub4RLogger$getClassName()
```

Returns: the class name

Method `getClass()`**:** Get class

Usage:

```
d4storagehub4RLogger$getClass()
```

Returns: the class

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

Usage:

```
d4storagehub4RLogger$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Note

Logger class used internally by d4storagehub4R

`StoragehubManager` *StoragehubManager*

Description

StoragehubManager

StoragehubManager

Value

Object of `R6Class` for modelling a D4Science StoragehubManager

Super class

`d4storagehub4R::d4storagehub4RLogger` -> `StoragehubManager`

Methods

Public methods:

- `StoragehubManager$new()`
- `StoragehubManager$getToken()`
- `StoragehubManager$getUserProfile()`
- `StoragehubManager$getUserWorkspace()`
- `StoragehubManager$fetchWSEndpoint()`

- StoragehubManager$fetchUserProfile()
- StoragehubManager$getWSRoot()
- StoragehubManager$getWSRootID()
- StoragehubManager$getWSItem()
- StoragehubManager$getWSItemID()
- StoragehubManager$getWSVREFolder()
- StoragehubManager$getWSVREFolderID()
- StoragehubManager$listWSItems()
- StoragehubManager$listWSItemsByPath()
- StoragehubManager$searchWSItem()
- StoragehubManager$searchWSItemID()
- StoragehubManager$createFolder()
- StoragehubManager$uploadFile()
- StoragehubManager$deleteItem()
- StoragehubManager$shareItem()
- StoragehubManager$unshareItem()
- StoragehubManager$downloadItem()
- StoragehubManager$downloadItemByPath()
- StoragehubManager$getPublicFileLinkByID()
- StoragehubManager$getPublicFileLink()
- StoragehubManager$clone()

Method `new()`**:** Method is used to instantiate the StoragehubManager.

Usage:

```
StoragehubManager$new(
  token,
  token_type = "gcube",
  logger = NULL,
  keyring_backend = "env"
)
```

Arguments:

`token` user access token

`token_type` token type, either 'gcube' (default) or 'jwt'

`logger` logger can be either NULL, "INFO" (with minimum logs), or "DEBUG" (for complete curl http calls logs)

`keyring_backend` keyring backend to use.it can be set to use a different backend for storing the D4science gcube token with **keyring** (Default value is 'env').

Method `getToken()`**:** Get token

Usage:

```
StoragehubManager$getToken()
```

Returns: the user access token

Method `getUserProfile()`**:** Get user profile

Usage:

```
StoragehubManager$getUserProfile()
```

Returns: the user profile

Method `getUserWorkspace()`**:** Get user workspace

Usage:

```
StoragehubManager$getUserWorkspace()
```

Returns: the user workspace root path

Method `fetchWSEndpoint()`**:** Fetches the workspace endpoint from the D4Science ICProxy service

Usage:

```
StoragehubManager$fetchWSEndpoint()
```

Method `fetchUserProfile()`**:** Fetches the user profile

Usage:

```
StoragehubManager$fetchUserProfile()
```

Method `getWSRoot()`**:** Get workspace root

Usage:

```
StoragehubManager$getWSRoot()
```

Returns: the workspace root, as `list`

Method `getWSRootID()`**:** Get workspace root ID

Usage:

```
StoragehubManager$getWSRootID()
```

Returns: the workspace root ID, as `character`

Method `getWSItem()`**:** Get workspace item given a `itemPath` in a parent folder

Usage:

```
StoragehubManager$getWSItem(
  parentFolderID = NULL,
  itemPath,
  showHidden = FALSE
)
```

Arguments:

`parentFolderID` parent folder ID

`itemPath` item path

`showHidden` show hidden files

Returns: the workspace item, `NULL` if no workspace item existing

Method `getWSItemID()`**:** Get workspace item ID given a `itemPath` in a parent folder

Usage:

```
StoragehubManager$getWSItemID(
  parentFolderID = NULL,
  itemPath,
  showHidden = FALSE
)
```

Arguments:

`parentFolderID` parent folder ID

`itemPath` item path

`showHidden` show hidden files

Returns: the workspace item ID, `NULL` if no workspace item existing

Method `getWSVREFolder()`**:** Get VRE Folder

Usage:

```
StoragehubManager$getWSVREFolder()
```

Returns: the VRE folder, as `list`

Method `getWSVREFolderID()`**:** Get VRE Folder ID

Usage:

```
StoragehubManager$getWSVREFolderID()
```

Returns: the VRE folder ID, as `character`

Method `listWSItems()`**:** Lists workspace items given a parentFolder ID

Usage:

```
StoragehubManager$listWSItems(parentFolderID = NULL, showHidden = FALSE)
```

Arguments:

`parentFolderID` parent folder ID

`showHidden` show hidden files

Returns: an object of class `data.frame`

Method `listWSItemsByPath()`**:** Lists workspace items given a folder path

Usage:

```
StoragehubManager$listWSItemsByPath(folderPath, showHidden = FALSE)
```

Arguments:

`folderPath` folder path where to list items

`showHidden` show hidden files

Returns: an object of class `data.frame`

Method `searchWSItem()`**:** Searches for a workspace item given a item path

Usage:

```
StoragehubManager$searchWSItem(
  itemPath,
  includeVreFolder = TRUE,
  showHidden = FALSE
)
```

Arguments:

`itemPath` path of the item

`includeVreFolder` search also in VRE folder

`showHidden` show hidden files

Returns: the item, `NULL` if nothing found

Method `searchWSItemID()`**:** Searches for a workspace item ID given a item path

Usage:

```
StoragehubManager$searchWSItemID(
  itemPath,
  includeVreFolder = TRUE,
  showHidden = FALSE
)
```

Arguments:

`itemPath` path of the item

`includeVreFolder` search also in VRE folder

`showHidden` show hidden files

Returns: the item, `NULL` if nothing found

Method `createFolder()`**:** Creates a folder, given a folder path, a folder name/description. By default `recursive = TRUE` meaning that a folder path matching nested folders will trigger all nested folders. Setting `recursive = FALSE`, the folder creation will work only if the folder path matches an existing folder. The `hidden` (default `FALSE`) argument can be used to set hidden folders on the workspace. Using `folderID`, `recursive` will be set to `FALSE`.

Usage:

```
StoragehubManager$createFolder(
  folderPath = NULL,
  folderID = NULL,
  name,
  description = "",
  hidden = FALSE,
  recursive = TRUE
)
```

Arguments:

`folderPath` parent folder path where to create the folder

`folderID` parent folder ID where to create the folder

`name` name of the folder

`description` description of the folder

`hidden` hidden, default is `FALSE`

`recursive` recursive, default is `TRUE`

Returns: the ID of the created folder

Method `uploadFile()`**:** Uploads a file to a folder (given a folder path). The argument `description` can be used to further describe the file to upload. The argument `archive` (default = FALSE) indicates the type of item (FILE or ARCHIVE) to be uploaded.

Usage:

```
StoragehubManager$uploadFile(
  folderPath = NULL,
  folderID = NULL,
  file,
  description = basename(file),
  archive = FALSE
)
```

Arguments:

`folderPath` folder path where to upload the file

`folderID` folder ID where to upload the file

`file` file to upload

`description` file description, default would be the file basename

`archive` archive, default is `FALSE`

Returns: the ID of the uploaded file

Method `deleteItem()`**:** Deletes an item given its path on the workspace

Usage:

```
StoragehubManager$deleteItem(itemPath, force = FALSE)
```

Arguments:

`itemPath` item path

`force` whether to force deletion, default is `FALSE`

Returns: `TRUE` if deleted, `FALSE` otherwise

Method `shareItem()`**:** Shares an item with users

Usage:

```
StoragehubManager$shareItem(itemPath, defaultAccessType, users)
```

Arguments:

`itemPath` item path

`defaultAccessType` access type to use for sharing, among 'WRITE_ALL', 'WRITE_OWNER', 'READ_ONLY', 'ADMINISTRATOR'

`users` one or more user names with whom the item has to be shared

Returns: `TRUE` if shared, `FALSE` otherwise

Method `unshareItem()`**:** unshare an item

Usage:

```
StoragehubManager$unshareItem(itemPath, users)
```

Arguments:

`itemPath` item path

`users` users

Returns: `TRUE` if unshared, `FALSE` otherwise

Method `downloadItem()`**:** Download item

Usage:

```
StoragehubManager$downloadItem(item = NULL, wd = NULL)
```

Arguments:

`item` item

`wd` working directory where to download the item

Method `downloadItemByPath()`**:** Download item by path

Usage:

```
StoragehubManager$downloadItemByPath(path, wd = NULL)
```

Arguments:

`path` path

`wd` working directory where to download the item

Method `getPublicFileLinkByID()`**:** Get public file link by ID

Usage:

```
StoragehubManager$getPublicFileLinkByID(pathID)
```

Arguments:

`pathID` file item ID

Returns: the public file URL

Method `getPublicFileLink()`**:** Get public file link

Usage:

```
StoragehubManager$getPublicFileLink(path)
```

Arguments:

`path` file path

Returns: the public file URL

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Note

Main user class to be used with **d4storagehub4R**

Deprecated

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
  manager <- StoragehubManager$new(
    token = "<your token>",
    logger = "DEBUG"
  )

## End(Not run)
```

Index