

Package ‘d4storagehub4R’

May 2, 2022

Version 0.3

Date 2022-05-02

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/ebondel/d4storagehub4R>,
<https://www.d4science.org/>, <https://dev.d4science.org/>

BugReports <https://github.com/ebondel/d4storagehub4R/issues>

LazyLoad yes

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation no

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

Repository CRAN

Date/Publication 2022-05-02 09:00:05 UTC

R topics documented:

d4storagehub4R	2
d4storagehub4RLogger	2
StoragehubManager	4

Index	10
--------------	-----------

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.blonde11@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:**

- [d4storagehub4RLogger\\$logger\(\)](#)
- [d4storagehub4RLogger\\$INFO\(\)](#)
- [d4storagehub4RLogger\\$WARN\(\)](#)
- [d4storagehub4RLogger\\$ERROR\(\)](#)
- [d4storagehub4RLogger\\$new\(\)](#)
- [d4storagehub4RLogger\\$getClassname\(\)](#)
- [d4storagehub4RLogger\\$getClass\(\)](#)
- [d4storagehub4RLogger\\$clone\(\)](#)

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	<i>StoragehubManager</i>
-------------------	--------------------------

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$fetchUserProfile()`
- `StoragehubManager$getWSRootID()`
- `StoragehubManager$getWSItemID()`
- `StoragehubManager$getWSElementID()`
- `StoragehubManager$listWSItems()`
- `StoragehubManager$listWSElements()`
- `StoragehubManager$listWSItemsByPath()`
- `StoragehubManager$listWSElementsByPath()`
- `StoragehubManager$searchWSItemID()`
- `StoragehubManager$searchWSFolderID()`
- `StoragehubManager$createFolder()`
- `StoragehubManager$uploadFile()`
- `StoragehubManager$deleteItem()`
- `StoragehubManager$shareItem()`
- `StoragehubManager$getPublicFileLink()`
- `StoragehubManager$clone()`

Method `new()`: Method is used to instantiate the [StoragehubManager](#). By default, the url is inherited through D4Science Iproxy service.

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 `getWSRootID()`: Get workspace root ID

Usage:

`StoragehubManager$getWSRootID()`

Returns: the workspace root ID, as character

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

Usage:

`StoragehubManager$getWSItemID(parentFolderID = NULL, folderPath)`

Arguments:

`parentFolderID` parent folder ID

`folderPath` folder path

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

Method `getWSElementID()`: Get workspace element ID given a `folderPath` in a parent folder

Usage:

`StoragehubManager$getWSElementID(parentFolderID = NULL, folderPath)`

Arguments:

`parentFolderID` parent folder ID

`folderPath` folder path

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

Method `listWSItems()`: Lists workspace items given a `parentFolderID`

Usage:

`StoragehubManager$listWSItems(parentFolderID = NULL)`

Arguments:

`parentFolderID` parent folder ID

Returns: an object of class `data.frame`

Method `listWSElements()`: Lists workspace elements given a `parentFolderID`

Usage:

```
StoragehubManager$listWSElements(parentFolderID = NULL)
```

Arguments:

parentFolderID parent folder ID

Returns: an object of class data.frame

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

Usage:

```
StoragehubManager$listWSItemsByPath(folderPath)
```

Arguments:

folderPath folder path where to list items

Returns: an object of class data.frame

Method listWSElementsByPath(): Lists workspace elements given a folder path

Usage:

```
StoragehubManager$listWSElementsByPath(folderPath)
```

Arguments:

folderPath folder path where to list elements

Returns: an object of class data.frame

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

Usage:

```
StoragehubManager$searchWSItemID(itemPath)
```

Arguments:

itemPath path of the item

Returns: the item ID, NULL if nothing found

Method searchWSFolderID(): Searches for a workspace element ID given a item path

Usage:

```
StoragehubManager$searchWSFolderID(folderPath)
```

Arguments:

folderPath path of the element

Returns: the item ID, 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.

Usage:

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

Arguments:

folderPath parent folder path 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,
  file,
  description = basename(file),
  archive = FALSE
)
```

Arguments:

folderPath folder path 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 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, use getWSItemID

Deprecated, use listWSItems

Deprecated, use listWSItemsByPath

Deprecated, use searchWSItemID

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

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

Index

- * **logger**
 - d4storagehub4RLogger, [2](#)
- * **manager**
 - StoragehubManager, [4](#)
- * **storagehub**
 - StoragehubManager, [4](#)

- d4storagehub4R, [2](#)
- d4storagehub4R-package
 - (d4storagehub4R), [2](#)
- d4storagehub4R::d4storagehub4RLogger,
[4](#)
- d4storagehub4RLogger, [2](#)

- R6Class, [2](#), [4](#)

- StoragehubManager, [4](#), [5](#)