

# Package ‘ocs4R’

March 17, 2022

**Version** 0.2-3

**Date** 2022-03-17

**Title** Interface to Open Collaboration Services (OCS) REST API

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

**Depends** R (>= 3.3.0), methods

**Imports** R6, openssl, curl, httr, jsonlite, XML, keyring

**Suggests** testthat

**Description** Provides an Interface to Open Collaboration Services 'OCS' (<<https://www.open-collaboration-services.org/>>) REST API.

**License** MIT + file LICENSE

**URL** <https://github.com/eblondel/ocs4R>

**BugReports** <https://github.com/eblondel/ocs4R/issues>

**LazyLoad** yes

**RoxygenNote** 7.1.0

**NeedsCompilation** no

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

**Repository** CRAN

**Date/Publication** 2022-03-17 19:40:03 UTC

## R topics documented:

ocs4R . . . . .	2
ocs4RLogger . . . . .	2
ocsApiSharingManager . . . . .	3
ocsApiUserProvisioningManager . . . . .	4
ocsApiWebdavManager . . . . .	5
ocsManager . . . . .	7
ocsRequest . . . . .	9

<b>Index</b>	<b>11</b>
--------------	-----------

---

ocs4R

*Interface to 'OCS' REST API*

---

### Description

Provides an Interface to 'OCS' (<<https://www.open-collaboration-services.org/>>) REST API.

### Details

Package: ocs4R  
Type: Package  
Version: 0.2-3  
Date: 2022-3-17  
License: MIT  
LazyLoad: yes

### Author(s)

Emmanuel Blondel <[emmanuel.blondel1@gmail.com](mailto:emmanuel.blondel1@gmail.com)>

---

ocs4RLogger

*ocs4RLogger*

---

### Description

ocs4RLogger

### 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

**Note**

Logger class used internally by ocs4R

---

ocsApiSharingManager *ocsApiSharingManager*

---

**Description**

ocsApiSharingManager

**Format**

[R6Class](#) object.

**Value**

Object of [R6Class](#) for modelling an ocsManager for the Sharing API

**General Methods (inherited from 'ocsManager')**

`new(url, user, pwd, logger, keyring_backend)` This method is used to instantiate an ocsApiSharingManager. The user/pwd are mandatory in order to connect to 'ocs'.

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

The keyring\_backend can be set to use a different backend for storing the user password with **keyring** (Default value is 'env').

`connect()` A method to connect to 'ocs' and set version/capabilities

`getVersion()` Get the 'ocs' server version

`getCapabilities()` Get the 'ocs' server capabilities

**'OCS' Share API methods**

`getShares(path, reshares, shared_with_me, state, subfiles, pretty)` Get list of shares as list. To return as data.frame, set pretty = TRUE. The method accepts additional parameters.

`createShare(path, name, shareType, shareWith, publicUpload, password, permissions = NULL, expireDate = NULL)`

Creates a share for the path (file or folder), given a name. The shareType should be among values 'user', 'group', 'publiclink' or 'federated'. The shareWith is required for shareType 'user' and 'group' and corresponds to the username or groupname. The permissions can be set among values 'read', 'update', 'create', 'delete', 'read-write', 'share', 'all'. By default the permissions will be the default permissions set by the ocs server (by default 'all').

`shareWithUser(path, name, username, permissions, pretty)` Shares a resource (file or folder) with a user given its username handled with argument username. The permissions can be set among values 'read', 'update', 'create', 'delete', 'read-write', 'share', 'all'. By default the permissions will be the default permissions set by the ocs server (by default 'all'). Returns the share properties as list (or as data.frame if pretty is set to TRUE).

shareWithGroup(path, name, group, permissions, pretty) Shares a resource (file or folder) with a group given its name handled with argument group. The permissions can be set among values 'read', 'update', 'create', 'delete', 'read-write', 'share', 'all'. By default the permissions will be the default permissions set by the ocs server (by default 'all'). Returns the share properties as list (or asdata.frame if pretty is set to TRUE).

shareAsPublicLink(path, name, publicUpload, password, permissions, expireDate) Shares a resource (file or folder) as public link. The function returns the public link generated by ocs.

### Author(s)

Emmanuel Blondel <emmanuel.blondell@gmail.com>

---

ocsApiUserProvisioningManager  
*ocsApiUserProvisioningManager*

---

### Description

ocsApiUserProvisioningManager

### Format

[R6Class](#) object.

### Value

Object of [R6Class](#) for modelling an ocsManager for Webdav API

### General Methods (inherited from 'ocsManager')

new(url, user, pwd, logger, keyring\_backend) This method is used to instantiate an ocsApiUserProvisioningManager. The user/pwd are mandatory in order to connect to 'ocs'

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

The keyring\_backend can be set to use a different backend for storing the user password with **keyring** (Default value is 'env').

connect() A method to connect to 'ocs' and set version/capabilities

getVersion() Get the 'ocs' server version

getCapabilities() Get the 'ocs' server capabilities

**User Provisioning API methods**

`addUser(userid, email, password, groups)` Adds a user given a `userid` (required). All other fields (`email`, `password`, `groups`) are optional for the user creation. Returns `TRUE` if the user is added, `FALSE` otherwise.

`getUsers()` Get the list of users. This method returns a vector of class `'character'` giving the user IDs available in the OCS cloud platform.

`getUser(userid, pretty)` Get the user details from its `userid`. If the argument `pretty` is set to `TRUE`, this will return an object of class `data.frame`, otherwise (by default) it returns an object of class `list`.

`editUser(userid, key, value)` Edits a user, identifier by a `userid`. The user property to be edited should be set using its key (eg `display`) and the value to be modified for this key. Returns `TRUE` if the user is edited, `FALSE` otherwise.

`editUserDisplayName(userid, displayName)` Edits a user display name.

`editUserEmail(userid, email)` Edits a user email.

`editUserPassword(userid, password)` Edits a user password.

`editUserQuota(userid, quota)` Edits a user quota.

`enableUser(userid)` Enables a user. Returns `TRUE` if enabled.

`disableUser(userid)` Disables a user. Returns `TRUE` if disabled.

`deleteUser(userid)` Deletes a user. Returns `TRUE` if deleted.

`getUserGroups(userid)` Get user group(s). This method returns a vector of class `'character'` giving the usergroups IDs

**`codeaddToGroup(userid, groupid)`** Adds a user to a group.

`removeFromGroup(userid, groupid)` Removes a user from a group.

`getGroups(search, limit, offset)` Get the list of groups. This method returns a vector of class `'character'` giving the usergroups IDs

`getGroup(groupid)` Get the group including member users from its `groupid`.

`addGroup(groupid)` Add group given a `groupid` (required).

`deleteGroup(groupid)` Deletes a group. Returns `TRUE` if deleted.

**Author(s)**

Emmanuel Blondel <emmanuel.blondell@gmail.com>

---

ocsApiWebdavManager    *ocsApiWebdavManager*

---

**Description**

ocsApiWebdavManager

**Format**

[R6Class](#) object.

**Value**

Object of [R6Class](#) for modelling an ocsManager for Webdav API

**General Methods (inherited from 'ocsManager')**

`new(url, user, pwd, logger, keyring_backend)` This method is used to instantiate an ocsApi-WebdavManager. The user/pwd are mandatory in order to connect to 'ocs'.

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

The `keyring_backend` can be set to use a different backend for storing the user password with **keyring** (Default value is 'env').

`connect()` A method to connect to 'ocs' and set version/capabilities

`getVersion()` Get the 'ocs' server version

`getCapabilities()` Get the 'ocs' server capabilities

**WebDAV methods**

`getWebdavRoot()` Get the 'ocs' WebDAV root URL

`listFiles(relPath)` WebDAV method to list folders/files given a relative path. The relative path is set to "/" by default, which corresponds to the root of the 'ocs' repository.

`makeCollection(name, relPath)` WebDAV method to make a collection. By default `relPath` is set to "/" (root). The name is the name of the new collection to be created. The function is recursive in the sense that a name can be provided as relative path of a collection tree (eg `newfolder1/newfolder2/newfolder3`), the function will create recursively as many collections are handled in the name.

`uploadFile(filename, relPath, delete_if_existing)` WebDAV method to upload a file. By default `relPath` is set to "/" (root).

`deleteFile(filename, relPath)` WebDAV method to delete a file. By default `relPath` is set to "/" (root).

`getPublicFile(share_token)` Get details of a shared public file given its share token

**Author(s)**

Emmanuel Blondel <emmanuel.blondell@gmail.com>

---

ocsManager

*ocsManager*

---

## Description

ocsManager

## Format

[R6Class](#) object.

## Value

Object of [R6Class](#) for modelling an ocsManager

## General Methods

`new(url, user, pwd, logger, keyring_backend)` This method is used to instantiate an ocsManager. The user/pwd are mandatory in order to connect to 'ocs'.

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

The keyring\_backend can be set to use a different backend for storing the user password with **keyring** (Default value is 'env').

`connect()` A method to connect to 'ocs' and set version/capabilities

`getVersion()` Get the 'ocs' server version

`getCapabilities()` Get the 'ocs' server capabilities

`getWebdavRoot()` Get the 'ocs' WebDAV root URL

## WebDAV methods

`listFiles(relPath)` WebDAV method to list folders/files given a relative path. The relative path is set to "/" by default, which corresponds to the root of the 'ocs' repository.

`makeCollection(name, relPath)` WebDAV method to make a collection. By default relPath is set to "/" (root). The name is the name of the new collection to be created. The function is recursive in the sense that a name can be provided as relative path of a collection tree (eg newfolder1/newfolder2/newfolder3), the function will create recursively as many collections are handled in the name.

`uploadFile(filename, relPath)` WebDAV method to upload a file. By default relPath is set to "/" (root).

**'OCS' Share API methods**

`getShares(path, reshares, shared_with_me, state, subfiles, pretty)` Get list of shares as list. To return as `data.frame`, set `pretty = TRUE`. The method accepts additional parameters.

`createShare(path, name, shareType, shareWith, publicUpload, password, permissions = NULL, expireDate = NULL)`  
Creates a share for the path (file or folder), given a name. The `shareType` should be among values `'user'`, `'group'`, `'publiclink'` or `'federated'`. The `shareWith` is required for `shareType` `'user'` and `'group'` and corresponds to the username or groupname. The permissions can be set among values `'read'`, `'update'`, `'create'`, `'delete'`, `'read-write'`, `'share'`, `'all'`. By default the permissions will be the default permissions set by the ocs server (by default `'all'`).

`shareWithUser(path, name, username, permissions, pretty)` Shares a resource (file or folder) with a user given its username handled with argument `username`. The permissions can be set among values `'read'`, `'update'`, `'create'`, `'delete'`, `'read-write'`, `'share'`, `'all'`. By default the permissions will be the default permissions set by the ocs server (by default `'all'`). Returns the share properties as list (or `asdata.frame` if `pretty` is set to `TRUE`).

`shareWithGroup(path, name, group, permissions, pretty)` Shares a resource (file or folder) with a group given its name handled with argument `group`. The permissions can be set among values `'read'`, `'update'`, `'create'`, `'delete'`, `'read-write'`, `'share'`, `'all'`. By default the permissions will be the default permissions set by the ocs server (by default `'all'`). Returns the share properties as list (or `asdata.frame` if `pretty` is set to `TRUE`).

`shareAsPublicLink(path, name, publicUpload, password, permissions, expireDate)` Shares a resource (file or folder) as public link. The function returns the public link generated by ocs.

**Author(s)**

Emmanuel Blondel <emmanuel.blondell@gmail.com>

**Examples**

```
## Not run:
#Not Run:
#Connect to an OCS API
OCS <- ocsManager$new(url = ocs_url, user = ocs_user, pwd = ocs_pwd, logger = "DEBUG")
version <- OCS$getVersion()
caps <- OCS$getCapabilities()

#OCS User Provisioning API
#-----
#users
users <- OCS$getUsers() #get users
user <- OCS$getUser("admin") #get a user
user.df <- OCS$getUser("admin", TRUE) #the same user as data.frame
added <- OCS$addUser("john.doe", password = "ocs4john") #add a user
disabled <- OCS$disableUser("john.doe") #disable a user
enabled <- OCS$enableUser("john.doe") #enable a user
edited <- OCS$editUser("john.doe", key = "display", value = "John Doe") #edit user
#edit some user field
edited2 <- OCS$editUserDisplayName("john.doe", displayName = "John Doe Jr.")
deleted <- OCS$deleteUser("john.doe")
```



```

#groups
admingroups <- OCS$getUserGroups("admin")
groups <- OCS$getGroups()
added <- OCS$addGroup("scientists") #add a new group
sc_group <- OCS$getGroup("scientists") #get group details
added <- OCS$addToGroup("john.doe", "scientists") #add user to group
removed <- OCS$removeFromGroup("john.doe", "scientists") #remove user from group
deleted <- OCS$deleteGroup("scientists")

#OCS Webdav API
#-----
#list files
files <- OCS$listFiles()
subfiles <- OCS$listFiles("Documents")
#make collection
OCS$makeCollection("myfolder")
subfiles <- OCS$listFiles("myfolder")
#upload a file?
filename <- "magic.txt"
file.create(filename); writelines("ocs4R is great", filename)
#we upload the file in 'Documents' folder
OCS$uploadFile(filename, "/Documents")
#check if file is uploaded
OCS$listFiles('Documents')

#OCS Sharing API
#-----
#let's add a user with User provisioning API
added <- OCS$addUser("john.doe", password = "ocs4john") #add a user
#let's share the previously uploaded file with John Doe
OCS$shareWithUser("/Documents", filename, "john.doe")

## End(Not run)

```

---

ocsRequest

*ocsRequest*

---

## Description

ocsRequest

## Format

[R6Class](#) object.

## Value

Object of [R6Class](#) for modelling a generic 'ocs' web-service request

**Methods**

`new(type, url, request, user, pwd, token, cookies, format, namedParams, content, contentType, filename, l`

This method is used to instantiate a object for doing an 'ocs' web-service request

`getRequest()` Get the request payload

`getRequestHeaders()` Get the request headers

`getStatus()` Get the request status code

`getResponse()` Get the request response

`getException()` Get the exception (in case of request failure)

`getResult()` Get the result TRUE if the request is successful, FALSE otherwise

**Note**

Abstract class used internally by **ocs4R**

**Author(s)**

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

# Index

- \* **api**
    - ocsApiSharingManager, [3](#)
    - ocsApiUserProvisioningManager, [4](#)
    - ocsApiWebdavManager, [5](#)
  - \* **logger**
    - ocs4RLogger, [2](#)
  - \* **manager**
    - ocsApiSharingManager, [3](#)
    - ocsApiUserProvisioningManager, [4](#)
    - ocsApiWebdavManager, [5](#)
    - ocsManager, [7](#)
  - \* **ocs**
    - ocsApiSharingManager, [3](#)
    - ocsApiUserProvisioningManager, [4](#)
    - ocsApiWebdavManager, [5](#)
    - ocsManager, [7](#)
    - ocsRequest, [9](#)
  - \* **request**
    - ocsRequest, [9](#)
  - \* **sharing**
    - ocsApiSharingManager, [3](#)
  - \* **userprovisioning**
    - ocsApiUserProvisioningManager, [4](#)
  - \* **webdav**
    - ocsApiWebdavManager, [5](#)
- ocs4R, [2](#)  
ocs4R-package (ocs4R), [2](#)  
ocs4RLogger, [2](#)  
ocsApiSharingManager, [3](#)  
ocsApiUserProvisioningManager, [4](#)  
ocsApiWebdavManager, [5](#)  
ocsManager, [7](#)  
ocsRequest, [9](#)
- R6Class, [2-4](#), [6](#), [7](#), [9](#)