Package 'ocs4R'

May 9, 2026

Version 0.3.1

Date 2026-04-29

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 7.3.1

NeedsCompilation no

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

Repository CRAN

Date/Publication 2026-04-29 22:10:02 UTC

Contents

ocs4R *Interface to 'OCS' REST API*

Description

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

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

See Also

Useful links:

- https://github.com/eblondel/ocs4R
- Report bugs at https://github.com/eblondel/ocs4R/issues

ocs4RLogger *ocs4RLogger*

Description

ocs4RLogger

ocs4RLogger

Format

R6Class object.

Value

Object of R6Class for modelling a simple logger

Public fields

`verbose.info` is info logger active?

`verbose.debug` is debug logger active?

`loggerType` logger type

Methods

Public methods:

- ocs4RLogger$new()
- ocs4RLogger$logger()
- ocs4RLogger$INFO()
- ocs4RLogger$WARN()
- ocs4RLogger$ERROR()
- ocs4RLogger$getClassName()
- ocs4RLogger$getClass()
- ocs4RLogger$clone()

Method `new()`**:** Initializes the logger

Usage:

```
ocs4RLogger$new(logger = NULL)
```

Arguments:

`logger` the type of logger. Default is `NULL`, accepts `INFO` or `DEBUG`

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

Usage:

```
ocs4RLogger$logger(type, text)
```

Arguments:

`type` type of log
`text` text

Method `INFO()`**:** Logger to report information. Used internally

Usage:

```
ocs4RLogger$INFO(text)
```

Arguments:

`text` text

Method `WARN()`**:** Logger to report warnings. Used internally

Usage:

```
ocs4RLogger$WARN(text)
```

Arguments:

`text` text

Method `ERROR()`**:** Logger to report errors Used internally

Usage:

```
ocs4RLogger$ERROR(text)
```

Arguments:

`text` text

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

Usage:

```
ocs4RLogger$getClassName()
```

Returns: the class name

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

Usage:

```
ocs4RLogger$getClass()
```

Returns: the class

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

Usage:

```
ocs4RLogger$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Note

Logger class used internally by ocs4R

`ocsApiSharingManager` *ocsApiSharingManager*

Description

ocsApiSharingManager

ocsApiSharingManager

Format

`R6Class` object.

Value

Object of `R6Class` for modelling an ocsManager for the Sharing API

Super classes

`ocs4R::ocs4RLogger` -> `ocs4R::ocsManager` -> `ocsApiSharingManager`

Methods

Public methods:

- ocsApiSharingManager$new()
- ocsApiSharingManager$getShares()
- ocsApiSharingManager$createShare()
- ocsApiSharingManager$shareWithUser()
- ocsApiSharingManager$shareWithGroup()
- ocsApiSharingManager$shareAsPublicLink()
- ocsApiSharingManager$clone()

Method `new()`**:** Initialize manager

Usage:

```
ocsApiSharingManager$new(url, user, pwd, logger = NULL, keyring_backend = NULL)
```

Arguments:

`url` url

`user` user

`pwd` pwd

`logger` logger

`keyring_backend` backend to use with **keyring**. Default is `NULL`

Method `getShares()`**:** Get list of shares as `list`. To return as `data.frame`, set `pretty = TRUE`. The method accepts additional parameters.

Usage:

```
ocsApiSharingManager$getShares(
  path = NULL,
  reshares = FALSE,
  shared_with_me = NULL,
  state = NULL,
  subfiles = FALSE,
  pretty = FALSE
)
```

Arguments:

`path` path

`reshares` reshares

`shared_with_me` list only those shared with me?

`state` state

`subfiles` subfiles

`pretty` pretty

Returns: the list of shares as `list` or `data.frame`

Method `createShare()`**:** 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').

Usage:

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

Arguments:

`path` path

`name` name

`shareType` the type of share

`shareWith` a list of users to share with

`publicUpload` public upload

`password` to set to access the share

`permissions` permissions

`expireDate` expire date

Method `shareWithUser()`**:** 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

Usage:

```
ocsApiSharingManager$shareWithUser(
  path,
  name,
  username,
  permissions = NULL,
  pretty = FALSE
)
```

Arguments:

`path` path

`name` name

`username` username

`permissions` permissions

`pretty` pretty

`group` group

Returns: the share properties as `list` (or as`data.frame` if `pretty` is set to TRUE).

Method `shareWithGroup()`**:** 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').

Usage:

```
ocsApiSharingManager$shareWithGroup(
  path,
  name,
  group,
  permissions = NULL,
  pretty = FALSE
)
```

Arguments:

`path` path

`name` name

`group` group

`permissions` permissions

`pretty` pretty

Returns: the share properties as `list` (or as`data.frame` if `pretty` is set to TRUE).

Method `shareAsPublicLink()`**:** Shares a resource (file or folder) as public link. The function returns the public link generated by ocs.

Usage:

```
ocsApiSharingManager$shareAsPublicLink(
  path,
  name = NULL,
  publicUpload = FALSE,
  password = NULL,
  permissions = NULL,
  expireDate = NULL
)
```

Arguments:

`path` path

`name` name

`publicUpload` public upload?

`password` password

`permissions` permissions

`expireDate` expire date

`return` the public sharing link

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

ocsApiUserProvisioningManager *ocsApiUserProvisioningManager*

Description

ocsApiUserProvisioningManager

ocsApiUserProvisioningManager

Format

R6Class object.

Value

Object of R6Class for modelling an ocsManager for Webdav API

Super classes

ocs4R::ocs4RLogger -> ocs4R::ocsManager -> ocsApiUserProvisioningManager

Methods

Public methods:

- ocsApiUserProvisioningManager$new()
- ocsApiUserProvisioningManager$addUser()
- ocsApiUserProvisioningManager$getUsers()
- ocsApiUserProvisioningManager$getUser()
- ocsApiUserProvisioningManager$editUser()
- ocsApiUserProvisioningManager$editUserDisplayName()
- ocsApiUserProvisioningManager$editUserEmail()
- ocsApiUserProvisioningManager$editUserPassword()
- ocsApiUserProvisioningManager$editUserQuota()
- ocsApiUserProvisioningManager$enableUser()
- ocsApiUserProvisioningManager$disableUser()
- ocsApiUserProvisioningManager$deleteUser()
- ocsApiUserProvisioningManager$getUserGroups()
- ocsApiUserProvisioningManager$addToGroup()
- ocsApiUserProvisioningManager$removeFromGroup()
- ocsApiUserProvisioningManager$createSubadmin()
- ocsApiUserProvisioningManager$removeSubadmin()
- ocsApiUserProvisioningManager$getSubadminGroups()
- ocsApiUserProvisioningManager$getGroups()
- ocsApiUserProvisioningManager$addGroup()

- ocsApiUserProvisioningManager$getGroup()
- ocsApiUserProvisioningManager$deleteGroup()
- ocsApiUserProvisioningManager$getSubadmins()
- ocsApiUserProvisioningManager$clone()

Method `new()`**:** Initialize manager

Usage:

```
ocsApiUserProvisioningManager$new(
  url,
  user,
  pwd,
  logger = NULL,
  keyring_backend = "env"
)
```

Arguments:

`url` url

`user` user

`pwd` pwd

`logger` logger

`keyring_backend` backend to use with **keyring**. Default is `NULL`

Method `addUser()`**:** 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.

Usage:

```
ocsApiUserProvisioningManager$addUser(
  userid,
  email = NULL,
  password = NULL,
  groups = NULL
)
```

Arguments:

`userid` user ID

`email` email

`password` user password

`groups` groups

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

Usage:

```
ocsApiUserProvisioningManager$getUsers()
```

Method `getUser()`**:** 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`.

Usage:

```
ocsApiUserProvisioningManager$getUser(userid, pretty = FALSE)
```

Arguments:

`userid` user ID

`pretty` pretty

Method `editUser()`**:** 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.

Usage:

```
ocsApiUserProvisioningManager$editUser(userid, key, value)
```

Arguments:

`userid` user ID

`key` key

`value` value

Method `editUserDisplayName()`**:** Edits a user display name.

Usage:

```
ocsApiUserProvisioningManager$editUserDisplayName(userid, displayName)
```

Arguments:

`userid` user ID

`displayName` display name

Method `editUserEmail()`**:** Edits a user email

Usage:

```
ocsApiUserProvisioningManager$editUserEmail(userid, email)
```

Arguments:

`userid` user ID

`email` email

Method `editUserPassword()`**:** Edits a user password

Usage:

```
ocsApiUserProvisioningManager$editUserPassword(userid, password)
```

Arguments:

`userid` user ID

`password` password

Method `editUserQuota()`**:** Edits a user quota

Usage:

```
ocsApiUserProvisioningManager$editUserQuota(userid, quota)
```

Arguments:

`userid` user ID

`quota` quota

Method `enableUser()`**:** Enables a user

Usage:

```
ocsApiUserProvisioningManager$enableUser(userid)
```

Arguments:

`userid` user ID

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

Method `disableUser()`**:** Disables a user

Usage:

```
ocsApiUserProvisioningManager$disableUser(userid)
```

Arguments:

`userid` user ID

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

Method `deleteUser()`**:** Deletes a user

Usage:

```
ocsApiUserProvisioningManager$deleteUser(userid)
```

Arguments:

`userid` user ID

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

Method `getUserGroups()`**:** Get user groups

Usage:

```
ocsApiUserProvisioningManager$getUserGroups(userid)
```

Arguments:

`userid` user ID

Returns: the user groups

Method `addToGroup()`**:** Adds a user to a group.

Usage:

```
ocsApiUserProvisioningManager$addToGroup(userid, groupid)
```

Arguments:

`userid` user ID

`groupid` group ID

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

Method `removeFromGroup()`**:** Removes a user from a group.

Usage:

```
ocsApiUserProvisioningManager$removeFromGroup(userid, groupid)
```

Arguments:

`userid` user ID

`groupid` group ID

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

Method `createSubadmin()`**:** Creates a subadmin

Usage:

```
ocsApiUserProvisioningManager$createSubadmin()
```

Method `removeSubadmin()`**:** Removes a subadmin

Usage:

```
ocsApiUserProvisioningManager$removeSubadmin()
```

Method `getSubadminGroups()`**:** Get subadmin groups

Usage:

```
ocsApiUserProvisioningManager$getSubadminGroups()
```

Method `getGroups()`**:** Get the list of groups. This method returns a vector of class 'character' giving the usergroups IDs

Usage:

```
ocsApiUserProvisioningManager$getGroups(
  search = NULL,
  limit = NULL,
  offset = NULL
)
```

Arguments:

`search` search

`limit` limit

`offset` offset

Method `addGroup()`**:** Adds a group

Usage:

```
ocsApiUserProvisioningManager$addGroup(groupid)
```

Arguments:

`groupid` group ID

Returns: `TRUE` if added, `FALSE`

Method `getGroup()`**:** Gets a group

Usage:

```
ocsApiUserProvisioningManager$getGroup(groupid)
```

Arguments:

`groupid` group ID

Returns: the group as `list` including the group ID and the list of users

Method `deleteGroup()`**:** Deletes a group

Usage:

```
ocsApiUserProvisioningManager$deleteGroup(groupid)
```

Arguments:

`groupid` group ID

Returns: `TRUE` if deleted, `FALSE`

Method `getSubadmins()`**:** Get subadmins

Usage:

```
ocsApiUserProvisioningManager$getSubadmins()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

`ocsApiWebdavManager` *ocsApiWebdavManager*

Description

ocsApiWebdavManager

ocsApiWebdavManager

Format

`R6Class` object.

Value

Object of `R6Class` for modelling an ocsManager for Webdav API

Super classes

`ocs4R::ocs4RLogger` -> `ocs4R::ocsManager` -> `ocsApiWebdavManager`

Methods

Public methods:

- ocsApiWebdavManager$new()
- ocsApiWebdavManager$getWebdavRoot()
- ocsApiWebdavManager$listFiles()
- ocsApiWebdavManager$makeCollection()
- ocsApiWebdavManager$uploadFile()
- ocsApiWebdavManager$deleteFile()
- ocsApiWebdavManager$downloadFile()
- ocsApiWebdavManager$getPublicFile()
- ocsApiWebdavManager$clone()

Method `new()`**:** Initialize manager

Usage:

```
ocsApiWebdavManager$new(url, user, pwd, logger = NULL, keyring_backend = NULL)
```

Arguments:

`url` url

`user` user

`pwd` pwd

`logger` logger

`keyring_backend` backend to use with **keyring**. Default is `NULL`

Method `getWebdavRoot()`**:** Get the 'ocs' WebDAV root URL

Usage:

```
ocsApiWebdavManager$getWebdavRoot()
```

Method `listFiles()`**:** 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.

Usage:

```
ocsApiWebdavManager$listFiles(relPath = "/")
```

Arguments:

`relPath` relative path

Returns: the list of files

Method `makeCollection()`**:** 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.

Usage:

```
ocsApiWebdavManager$makeCollection(name, relPath = "/")
```

Arguments:

`name` name

`relPath` relative path

Method `uploadFile()`**:** WebDAV method to upload a file. By default `relPath` is set to `"/"` (root).

Usage:

```
ocsApiWebdavManager$uploadFile(
  filename,
  relPath = "/",
  delete_if_existing = FALSE
)
```

Arguments:

`filename` file name

`relPath` relative path

`delete_if_existing` delete if existing file? Default is `FALSE`

Method `deleteFile()`**:** WebDAV method to delete a file. By default `relPath` is set to `"/"` (root).

Usage:

```
ocsApiWebdavManager$deleteFile(filename, relPath = "/")
```

Arguments:

`filename` file name

`relPath` relative path

Method `downloadFile()`**:** Downloads a file

Usage:

```
ocsApiWebdavManager$downloadFile(relPath, filename, outdir = ".")
```

Arguments:

`relPath` relative path

`filename` file name

`outdir` the out directory where to download the file

Method `getPublicFile()`**:** Get details of a shared public file given its share token

Usage:

```
ocsApiWebdavManager$getPublicFile(share_token)
```

Arguments:

`share_token` the share token

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

ocsManager *ocsManager*

Description

ocsManager

ocsManager

Format

R6Class object.

Value

Object of R6Class for modelling an ocsManager

Super class

ocs4R::ocs4RLogger -> ocsManager

Public fields

apis list of APIs

Methods

Public methods:

- ocsManager$new()
- ocsManager$connect()
- ocsManager$getVersion()
- ocsManager$getCapabilities()
- ocsManager$getAPIWebdavManager()
- ocsManager$getAPISharingManager()
- ocsManager$getAPIUserProvisioningManager()
- ocsManager$clone()

Method new(): 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 NULL, meaning the password is stored as private field).

Usage:

```
ocsManager$new(url, user, pwd, logger = NULL, keyring_backend = NULL)
```

Arguments:

url url

`user` user
`pwd` pwd
`logger` logger type
`keyring_backend` keyring back-end. Default is `NULL`

Method `connect()`**:** Method to connect to 'ocs' and set version/capabilities

Usage:

```
ocsManager$connect()
```

Method `getVersion()`**:** Get the 'ocs' server version

Usage:

```
ocsManager$getVersion()
```

Method `getCapabilities()`**:** Get the 'ocs' server capabilities

Usage:

```
ocsManager$getCapabilities()
```

Method `getAPIWebdavManager()`**:** Get the Webdav API manager

Usage:

```
ocsManager$getAPIWebdavManager()
```

Returns: an instance of ocsApiWebdavManager

Method `getAPISharingManager()`**:** Get the Sharing API manager

Usage:

```
ocsManager$getAPISharingManager()
```

Returns: an instance of ocsApiSharingManager

Method `getAPIUserProvisioningManager()`**:** Get the User Provisioning API manager

Usage:

```
ocsManager$getAPIUserProvisioningManager()
```

Returns: an instance of ocsApiUserProvisioningManager

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@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 auser
   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

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

Super class

ocs4R::ocs4RLogger -> ocsRequest

Methods

Public methods:

- ocsRequest$new()
- ocsRequest$execute()
- ocsRequest$getRequest()
- ocsRequest$getRequestHeaders()
- ocsRequest$getStatus()
- ocsRequest$getResponse()

- ocsRequest$getException()
- ocsRequest$getResult()
- ocsRequest$setResult()
- ocsRequest$clone()

Method `new()`**:** This method is used to instantiate a object for doing an 'ocs' web-service request

Usage:

```
ocsRequest$new(
  type,
  url,
  request,
  user = NULL,
  pwd = NULL,
  token = NULL,
  cookies = NULL,
  format = "json",
  namedParams = list(),
  content = NULL,
  contentType = "text/plain",
  filename = NULL,
  logger = NULL,
  ...
)
```

Arguments:

`type` type of request

`url` url

`request` request

`user` user

`pwd` pwd

`token` token

`cookies` cookies

`format` format. Default is "json"

`namedParams` a list of named parameters

`content` content

`contentType` content type. Default is "text/plain"

`filename` file name

`logger` logger

`...` additional parameters

Method `execute()`**:** Executes the request

Usage:

```
ocsRequest$execute()
```

Method `getRequest()`**:** Get request

Usage:

```
ocsRequest$getRequest()
```

Method `getRequestHeaders()`**:** Get request headers

Usage:

```
ocsRequest$getRequestHeaders()
```

Method `getStatus()`**:** Get status

Usage:

```
ocsRequest$getStatus()
```

Method `getResponse()`**:** Get response

Usage:

```
ocsRequest$getResponse()
```

Method `getException()`**:** Get exception

Usage:

```
ocsRequest$getException()
```

Method `getResult()`**:** Get result

Usage:

```
ocsRequest$getResult()
```

Method `setResult()`**:** Set result

Usage:

```
ocsRequest$setResult(result)
```

Arguments:

`result` result

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Note

Abstract class used internally by **ocs4R**

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Index