Package 'ndexr'

November 22, 2024

Type Package

Title NDEx R client library

Version 1.29.0

Date 2018-04-27

Description This package offers an interface to NDEx servers, e.g. the public server at http://ndexbio.org/. It can retrieve and save networks via the API. Networks are offered as RCX object and as igraph representation.

License BSD_3_clause + file LICENSE

Depends RCX

Imports httr, jsonlite, plyr, tidyr

RoxygenNote 7.2.3

Encoding UTF-8

biocViews Pathways, DataImport, Network

Suggests BiocStyle, testthat, knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/frankkramer-lab/ndexr

BugReports https://github.com/frankkramer-lab/ndexr/issues

git_url https://git.bioconductor.org/packages/ndexr

git_branch devel

git_last_commit b913e3a

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-11-21

Author Florian Auer [cre, aut] (ORCID:
<https://orcid.org/0000-0002-5320-8900>),
Frank Kramer [ctb],
Alex Ishkin [ctb],
Dexter Pratt [ctb]

Maintainer Florian Auer <florian.auer@informatik.uni-augsburg.de>

Contents

`ndexr-package` *NDEx R client library*

Description

The ndexr package offers an interface to NDEx servers, e.g. the public server at http://ndexbio.org/. It can retrieve and save networks via the API. Networks are offered as RCX objects.

Details

Package:	ndexr
Type:	Package
Version:	1.19.2
Date:	2016-12-02
License:	BSD_3_clause

Author(s)

Florian Auer <florian.auer@informatik.uni-augsburg.de>

Zaynab Hammoud <zaynab.hammoud@informatik.uni-augsburg.de>

Frank Kramer <frank.kramer@informatik.uni-augsburg.de>

Examples

```
## Not run:
require(ndexr)
## login to the NDEx server
ndexcon <- ndex_connect("username", "password")

## search the networks for "EGFR"
networks <- ndex_find_networks(ndexcon, "EGFR")
head(networks, 3)
```

```
## UUID of the first search result
networkId <- networks[1,'externalId']
networkId

## get summary of the network
ndex_network_get_summary(ndexcon, networkId)

## get the entire network as RCX object
rcx <- ndex_get_network(ndexcon, networkId)

## show the content (aspects) of the network
rcx$metaData

## visualize the network with RCX
RCX::visualize(rcx)

## upload network as a new network to the NDEx server
networkId <- ndex_create_network(ndexcon, rcx)

## do some other fancy stuff with the network, then
## update the network on the server
networkId <- ndex_update_network(ndexcon, rcx)

## realize, you did bad things to the poor network, so better
## delete it on the server
ndex_delete_network(ndexcon, networkId)

## End(Not run)
```

`listToRCode` *Translates a nested list (as provided by a yaml file) into R code defining the lists*

Description

Translates a nested list (as provided by a yaml file) into R code defining the lists

Usage

```
listToRCode(obj, indent = "    ", indentShift = "")
```

Arguments

`obj`	list; Nested list to be translated. The allowed list elements are: list, character, numeric, logical
`indent`	character (optional) (default: ' '[4 whitespaces]); Character(s) used for the indentation. Default: <tab>
`indentShift`	character (optional) (default: ''); Shifts the block to the right by putting the characters before each line (mostly used in internal recursion)

Value

character; R code for generating the nested list

Note

only for package maintenance!

Examples

```
test = list(bla='some text',blubb=list(a='more text', version='2.0'),justANumber=123456)
#$bla
#[1] "some text"
#$blubb
#$blubb$a
#[1] "more text"
#$blubb$version
#[1] "2.0"
#$justANumber
#[1] 123456
#
#listToRCode(test)
#[1] "list(\n\tbla=\"some text\",\n\tblubb=list(\n\t\ta=\"more text\",\n\t\tversion=\"2.0\"\n\t)\n)"
```

---

ndex_config *NDEx server api configuration*

Description

This nested list contains the url and methods for accessing the NDEx server via its REST full api. It contains specifications for NDEx server api version 1.3 and 2.0. The default api is specified by 'defaultVersion'. If possible, the version 2.0 should be used. Own configurations must contain a 'version' entry!

Usage

```
ndex_config
```

Format

An object of class `list` of length 4.

Value

Nested list resembling the NDEx server REST API structure

Examples

```
names(ndex_config$Version_2.0)
```

ndex_conf_header *Default header for the ndex_api_config.r file*

Description

Default header for the ndex_api_config.r file

Usage

```
ndex_conf_header
```

Format

An object of class `character` of length 1.

Value

character containing the header

Note

only for package maintenance!

Examples

```
NULL
```

ndex_connect *Connect to NDEx REST API*

Description

This function creates an NDExConnection which stores options and authentication details. It is a parameter required for most of the other ndexr functions. If username and password are missing an anonymous connection is created, which already offers most of the retrieval functionality.

Usage

```
ndex_connect(
  username,
  password,
  host = ndexConf$connection$host,
  apiPath = ndexConf$connection$api,
  ndexConf = ndex_config,
  verbose = FALSE
)
```

Arguments

username	character (optional); username
password	character (optional); password
host	character (default: ndexConf$connection$host); Host address of NDEx server; By default the url set in ndexConf$defaults$connection$host is used. ("http://www.ndexbio.org")
apiPath	character (default: ndexConf$connection$api); URL path of the REST api; By default the url set in ndexConf$defaults$connection$api is used. ("/v2" for NDEx version 2.0, "/rest" for NDEx version 1.3)
ndexConf	config object (nested list, default: ndex_config); Configuration of NDEx REST server; Set in ndex_config (set in ndex_api_config.r or ndex_api_config.yml): It contains specifications for NDEx server api version 1.3 and 2.0. The default api is specified by 'defaultVersion'
verbose	logical (optional); whether to print out extended feedback

Value

returns object of class NDExConnection which stores options, authentication and api configuration if successfull, NULL otherwise

See Also

ndex_config

Examples

```
## log in anonymously
ndexcon = ndex_connect()
## same as above with extended feedback
ndexcon = ndex_connect(verbose=TRUE)
## Not run:
## log in with credentials
ndexcon = ndex_connect('user','password')
## running some NDEx server locally
ndexcon = ndex_connect(host='localhost:8765')
## manually change the api and connection configuration
ndexcon = ndex_connect(ndexConf=ndex_config$Version_2.0)

## End(Not run)
```

---

ndex_create_group *Create Group*

Description

Create a group owned by the authenticated user based on the supplied group JSON object.

Usage

```
ndex_create_group(ndexcon, groupName, image, website, description, properties)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
groupName	character; name of the new graoup
image	character (optional); URL of the account owner's image
website	character (optional); URL of the account owner's web site
description	character (optional); Short description of this user
properties	list (optional); additional properties for the group

Value

url (including the UUID) of the newly created group

REST query

POST: ndex_configapigroup$create

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
# groupURL = ndex_create_group(ndexcon, 'SomeGroupName')
## [1] "http://public.ndexbio.org/v2/group/aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
# groupURL = ndex_create_group(ndexcon, 'SomeGroupName',
#                              image='http://bit.ly/1M3NoQZ',
#                              website='www.gidf.com',
#                              description='A very special group..')
NULL
```

---

ndex_create_network *Create a Network at a server from RCX data*

Description

This method creates a new network on the NDEx server from the given RCX object

Usage

```
ndex_create_network(ndexcon, rcx)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
rcx	RCX-object object

Details

Note: In future 'ndexr' uses the RCX-object from the corresponding package to handle the networks!

Value

UUID of the newly created network

REST query

POST (multipart/form-data): ndex_configapinetwork$create$url data: CXNetworkStream = data

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 1.3 and 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find a network and get its UUID
# networks = ndex_find_networks(ndexcon,"p53", "nci-pid")
# networkId = networks[1,"externalId"]
## Get the network data
# rcx = ndex_get_network(ndexcon, networkId)
## Do some changes to rcx..
## and create a new network
# networkId = ndex_create_network(ndexcon, rcx)
NULL
```

---

ndex_create_user *Create a user*

Description

Create a new user based on a JSON object specifying username, password, and emailAddress. Username and emailAddress must be unique in the database. If email verification is turned on on the server, this call returns code 220 (Accepted), the location field in the header has the URL to check the status of the newly created user account. If email verification is turned on off on the server, this function returns 201 (Created). The URL for getting the newly created user is in the response body and the Location header.

Usage

```
ndex_create_user(
  ndexcon,
  userName,
  password,
  emailAddress,
  isIndividual = TRUE,
  displayName,
  firstName,
  lastName,
  image,
  website,
  description,
  verbose = FALSE
)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userName	character; name of the new user
password	character; password for the new user
emailAddress	character (optional); email address (used for verification if enabled)
isIndividual	boolean (default: TRUE); True if this account is for an individual user. False means this account is for an organization or a project etc.
displayName	character (optional); Display name of this account, only applied to non-individual accounts.
firstName	character (optional); Account owner's first name, only applies to individual accounts.
lastName	character (optional); Account owner's last name, only appliies to individual accounts.
image	character (optional); URL of the account owner's image.
website	character (optional); URL of the account owner's web site
description	character (optional); Short description of this user.
verbose	logical (optional); whether to print out extended feedback

Value

UUID of the newly created user if email verification is turned off, else an empty string ("")

REST query

GET: ndex_configapiuser$create

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Create a new user
# userId = ndex_create_user(ndexcon, 'SomeUserName', 'SecretPassword', 'SomeUserName@ndex.org')
## [1] "uuuuuuuu-ssss-eeee-rrrr-123456789abc"
# userId = ndex_create_user(ndexcon, 'ASpecialProject', 'SecretPassword',
#                           'ASpecialProject@ndex.org', isIndividual=TRUE,
#                           displayName='Area51', firstName='John', lastName='Doe',
#                           website='www.gidf.com', description='Nothing to see here..')
NULL
```

`ndex_delete_group` *Delete Group*

Description

Delete the group specified by groupId

Usage

```
ndex_delete_group(ndexcon, groupId)
```

Arguments

`ndexcon`	object of class NDExConnection linkndex_connect
`groupId`	character; unique ID (UUID) of the group

Value

NULL if successfull, else an error is thrown

REST query

DELETE: ndex_configapigroup$delete

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find user and get its id
# user = ndex_find_user_byName(ndexcon, 'MyAccountName')
# userId = user$externalId
## Find the user's groups and get one group id
# groups = ndex_user_list_groups(ndexcon, userId)
# groupId = groups[1,"externalId"]
#ndex_delete_group(ndexcon,groupId)
NULL
```

---

ndex_delete_network *Delete a network*

Description

Delete a network

Usage

```
ndex_delete_network(ndexcon, networkId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
networkId	unique ID of the network

Value

NULL on success; Error else

REST query

DELETE: ndex_configapinetwork$delete

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 1.3 and 2.0

Examples

```
## Establish a server connections with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find a network and get its UUID
# networks = ndex_find_networks(ndexcon,"p53", "nci-pid")
# networkId = networks[1,"externalId"]
## Delete the network
# ndex_delete_network(ndexcon, networkId)
NULL
```

---

ndex_delete_user *Delete User*

Description

Deletes the authenticated user, removing any other objects in the database that depend on the user

Usage

```
ndex_delete_user(ndexcon, userId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userId	character; unique ID (UUID) of the user

Value

NULL if successfull, else an error is thrown

REST query

GET: ndex_configapiuser$delete

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find user and get its id
# user = ndex_find_user_byName(ndexcon, 'SomeUserName')
# userId = user$externalId
## Delete user
# ndex_delete_user(ndexcon, userId)
NULL
```

ndex_find_groups *Search groups in NDEx*

Description

Returns a SearchResult object which contains an array of Group objects

Usage

```
ndex_find_groups(ndexcon, searchString = "", start, size)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
searchString	string by which to search
start	integer (optional); specifies that the result is the nth page of the requested data. The default value is 0
size	integer (optional); specifies the number of data items in each page. The default value is 100

Value

Data frame with group information; NULL if no groups are found.

REST query

GET: ndex_configapisearch$user

Note

Compatible to NDEx server version 1.3 and 2.0

Search strings may be structured

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find a group
groups = ndex_find_groups(ndexcon,"Ideker Lab")
names(groups)
## [1] "properties"  "groupName"  "image"             "website"       "description"
## [6] "externalId"  "isDeleted"  "modificationTime"  "creationTime"
```

ndex_find_networks *Search networks in NDEx (by description)*

Description

This functions searches the public networks on an NDEx server for networks containing the supplied search string. T his search can be limited to certain accounts as well as in length.

Usage

```
ndex_find_networks(ndexcon, searchString = "", accountName, start, size)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
searchString	string by which to search
accountName	string (optional); constrain search to networks administered by this account
start	integer (optional); specifies that the result is the nth page of the requested data. The default value is 0
size	integer (optional); specifies the number of data items in each page. The default value is 100

Value

Data frame with network information: ID, name, whether it is public, edge and node count; source and format of network. NULL if no networks are found.

REST query

GET: ndex_configapisearch$network$search

Note

Compatible to NDEx server version 1.3 and 2.0

Search strings may be structured

Examples

```
ndexcon = ndex_connect()
networks = ndex_find_networks(ndexcon,"p53")
```

`ndex_find_users` *Search user in NDEx*

Description

Returns a SearchResult object which contains an array of User objects

Usage

```
ndex_find_users(ndexcon, searchString = "", start, size)
```

Arguments

`ndexcon`	object of class NDExConnection linkndex_connect
`searchString`	string by which to search
`start`	integer (optional); specifies that the result is the nth page of the requested data. The default value is 0
`size`	integer (optional); specifies the number of data items in each page. The default value is 100

Value

Data frame with user information; NULL if no user are found.

REST query

GET: ndex_configapisearch$user

Note

Compatible to NDEx server version 1.3 and 2.0

Search strings may be structured

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find a user
users = ndex_find_users(ndexcon,"ndextutorials")
names(users)
## [1] "properties"    "displayName"       "isIndividual"  "userName"     "password"
## [6] "isVerified"    "firstName"         "lastName"      "diskQuota"    "diskUsed"
##[11] "emailAddress"  "image"             "website"       "description"  "externalId"
##[16] "isDeleted"     "modificationTime"  "creationTime"
```

ndex_find_user_byId *Get User By UUID*

Description

Get User By UUID

Usage

```
ndex_find_user_byId(ndexcon, userId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userId	character; unique ID (UUID) of the user

Value

list of properties describing the user (externalId, emailAddress, website, etc.). Throws error (404) if user isn't found!

REST query

GET: ndex_configapiusergetbyId

Note

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find user by name
user = ndex_find_user_byName(ndexcon, 'ndextutorials')
## Find user by Id
user = ndex_find_user_byId(ndexcon, user$externalId)
```

ndex_find_user_byName *Get User By Name*

Description

Get User By Name

Usage

```
ndex_find_user_byName(ndexcon, name)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
name	name of the user

Value

list of properties describing the user (externalId, emailAddress, website, etc.). Throws error (404) if user isn't found!

REST query

GET: ndex_configapiusergetbyName

Note

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find user by name
user = ndex_find_user_byName(ndexcon, 'ndextutorials')
```

ndex_get_group *Get a Group*

Description

Get a Group

Usage

```
ndex_get_group(ndexcon, groupId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
groupId	character; unique ID (UUID) of the group

Value

list of properties describing the group (externalId, emailAddress, website, etc.). Throws error (404) if group isn't found!

REST query

GET: ndex_configapigroup$get

Note

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find a group
groups = ndex_find_groups(ndexcon,"Ideker Lab")
groupId = groups[1,"externalId"]
## Get group information
group = ndex_get_group(ndexcon, groupId)
```

---

ndex_get_network *Get complete network*

Description

Returns the specified network as RCX-object.

Usage

```
ndex_get_network(ndexcon, networkId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
networkId	unique ID of the network

Details

Note: In future 'ndexr' uses the RCX-object from the corresponding package to handle the networks!

This is performed as a monolithic operation, so it is typically advisable for applications to first use the getNetworkSummary method to check the node and edge counts for a network before retrieving the network. Uses getEdges (this procedure will return complete network with all elements) Nodes use primary ID of the base term ('represents' element) Edges use primary ID of the base term ('predicate', or 'p' element) Mapping table for the nodes is retrieved ('alias' and 'related' terms) to facilitate conversions/data mapping

Value

`RCX-object` object

REST query

GET: ndex_configapinetwork$get

Note

Compatible to NDEx server version 1.3 and 2.0

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find a network and get its UUID
networks = ndex_find_networks(ndexcon,"p53", "nci-pid")
networkId = networks[1,"externalId"]
## Get the network data
rcx = ndex_get_network(ndexcon, networkId)
```

---

ndex_group_delete_membership

Remove a Group Member

Description

Removes the member from the group

Usage

```
ndex_group_delete_membership(ndexcon, groupId, userId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
groupId	character; unique ID (UUID) of the group
userId	character; unique ID (UUID) of the user

Value

Empty string ("") on success, else error

REST query

DELETE: ndex_configapiuser$membership$delete

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find user and get own id
# user = ndex_find_user_byName(ndexcon, 'MyAccountName')
# userId = user$externalId
## Find own groups and get one group id
# groups = ndex_user_list_groups(ndexcon, userId)
# groupId = groups[1,"externalId"]
## Find an other user of the group and get the id
# users = ndex_group_list_users(ndexcon, groupId)
## Choose one user
# userId = users[1,"externalId"]
## Remove user from the group
# ndex_group_delete_membership(ndexcon, groupId, userId)
NULL
```

---

ndex_group_list_networks

Get Network Permissions of a Group

Description

Get Network Permissions of a Group

Usage

```
ndex_group_list_networks(
  ndexcon,
  groupId,
  permission = NULL,
  start = NULL,
  size = NULL
)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
groupId	character; unique ID (UUID) of the group
permission	character (optional) ("WRITE"|"READ) (default: "READ"); constrains the type of the returned permission.
start	integer (optional); specifies that the result is the nth page of the requested data.
size	integer (optional); specifies the number of data items in each page.

Value

List of network permissions of that group or empty object

REST query

GET: ndex_configapigroup$network$list

Note

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find a group
groups = ndex_find_groups(ndexcon,"Ideker Lab")
groupId = groups[1,"externalId"]
## List networks of the group
networks = ndex_group_list_networks(ndexcon, groupId)
networks = ndex_group_list_networks(ndexcon, groupId, permission='READ', start=0, size=10)
```

---

ndex_group_list_users *Get Members of a Group*

Description

Get Members of a Group

Usage

```
ndex_group_list_users(ndexcon, groupId, type = NULL, start = NULL, size = NULL)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
groupId	character; unique ID (UUID) of the group
type	character (optional); constrains the type of the returned membership. If not set (or NULL), all permission types will be returned.
start	integer (optional); specifies that the result is the nth page of the requested data.
size	integer (optional); specifies the number of data items in each page.

Value

List of permissions of that group or empty object

REST query

GET: ndex_configapigroup$membership$get

Note

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find a group
groups = ndex_find_groups(ndexcon,"Ideker Lab")
groupId = groups[1,"externalId"]
## Find other users of the group
# users = ndex_group_list_users(ndexcon, groupId)
# users = ndex_group_list_users (ndexcon, groupId, type='ADMIN', start=0, size=10)
```

---

ndex_group_network_get_permission

Get Group Permission for a Specific Network

Description

Get Group Permission for a Specific Network

Usage

```
ndex_group_network_get_permission(ndexcon, groupId, networkId)
```

Arguments

ndexcon	object of class NDExConnection ndex_connect
groupId	character; unique ID (UUID) of the group
networkId	character; unique ID (UUID) of the network

Value

Network permissions of that group or empty object

REST query

GET: ndex_configapigroup$network$get

Note

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find a group
groups = ndex_find_groups(ndexcon,"Ideker Lab")
groupId = groups[1,"externalId"]
## List networks of the group
networks = ndex_group_list_networks(ndexcon, groupId)
networkId = names(networks)[1]
## Get group's permission to the network
#group = ndex_group_network_get_permission(ndexcon, groupId, networkId)
```

---

ndex_group_set_membership

Add or Update a Group Member

Description

Updates the membership corresponding to the GroupMembership type specified in the URL parameter.

Usage

```
ndex_group_set_membership(ndexcon, groupId, userId, type = "MEMBER")
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
groupId	character; unique ID (UUID) of the group
userId	character; unique ID (UUID) of the user
type	character (optional)("GROUPADMIN"|"MEMBER")(default: "MEMBER"); Type of group membership

Value

Empty string ("") on success, else error

REST query

PUT: ndex_configapiuser$membership$update

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find user and get own id
# user = ndex_find_user_byName(ndexcon, 'MyAccountName')
# userId = user$externalId
## Find own groups and get one group id
# groups = ndex_user_list_groups(ndexcon, userId)
# groupId = groups[1,"externalId"]
## Find an other user and get the id
# user = ndex_find_user_byName(ndexcon, 'SomeOtherAccountName')
# userId = user$externalId
## Add other user to the group
# ndex_group_set_membership(ndexcon, groupId, userId)
## Update other user's group permission
# ndex_group_set_membership(ndexcon, groupId, userId, type='MEMBER')    ## same as before
## Make other user to group admin (lose own admin permission)
# ndex_group_set_membership(ndexcon, groupId, userId, type='GROUPADMIN')
NULL
```

---

ndex_helper_encodeParams

Adds Parameters to an url

Description

Encodes a given parameter within the url accordingly to the parameter configuration for the api.

Usage

```
ndex_helper_encodeParams(url, params, ...)
```

Arguments

url	character
params	(nested) list; "params" section of a api function definition in the api configuration (See ndex_config)
...	parameters defined by name used in the config

Details

The single parameter definitions are given as list by the "params" parameter. Each parameter is defined by a method, and, if applicable, a tag, a default value and/or an optional flag. There are three keywords defining the method: replace, append or parameter.

replace: The String defined by "tag" can be found within the url and will be replaced by the given value of the parameter. E.g. the tag "#NETWORKID#" in the url "/network/#NETWORKID#/provenance" is replaced by a value (e.g. "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee") given as network id, which leads to the url "/network/aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee/provenance".

append: The given value of the parameter is appended to an url. Therefore the order of the parameters in the params definition is used. E.g. the url "/network/search" and the given values for "start" = 0 and "size" = 100 generates the following url: "/network/search/0/100"

parameter: Encodes the given parameters as url parameter using the specified tag as parameter descriptor. E.g. a parameter with the tag "username" and the value "SomeName" is encoded in the url "/user" as follows: "/user?username=SomeName"

It is also possible to set parameter as optional (except for replace), or define default values. Values are assigned to the parameters using the parameter name in the ... parameter.

Value

URL with encoded parameters as character

Note

This function is internal.

Examples

```
## replace
url = "http://en.wikipedia.org/#NETWORKID#/index.php"
params = list(    network=list(    tag="#NETWORKID#", method="replace"))
values = c(network='aaaa-bb-cc-dddddd', bla='This is not used!')
ndexr:::ndex_helper_encodeParams(url, params=params, values)
## "http://en.wikipedia.org/aaaa-bb-cc-dddddd/index.php"

params = list( network=list( tag="#NETWORKID#", method="replace", default="xxxx-xx-xx-xxxxxx"))
values = c(bla='This is not used!')
ndexr:::ndex_helper_encodeParams(url, params=params, values)
## "http://en.wikipedia.org/xxxx-xx-xx-xxxxxx/index.php"

## parameter
url = "http://en.wikipedia.org/w/index.php"
params = list(    network=list(    tag="network", method="parameter"))
values = c(network='aaaa-bb-cc-dddddd', bla='This is not used!')
ndexr:::ndex_helper_encodeParams(url, params=params, values)
## "http://en.wikipedia.org/w/index.php?network=aaaa-bb-cc-dddddd"

values = c(bla='This is not used!')
params = list(    network=list(    tag="network", method="parameter", optional=TRUE))
ndexr:::ndex_helper_encodeParams(url, params=params, values)
```

```
## "http://en.wikipedia.org/w/index.php"

params = list( network=list( tag="network", method="parameter", default="xxxx-xx-xx-xxxxxx"))
ndexr:::ndex_helper_encodeParams(url, params=params, values)
## "http://en.wikipedia.org/w/index.php?network=xxxx-xx-xx-xxxxxx"

ndexr:::ndex_helper_encodeParams(url, params=params, values)
values = c(network='aaaa-bb-cc-dddddd', bla='This is not used!')
## "http://en.wikipedia.org/w/index.php?network=aaaa-bb-cc-dddddd"

## append
url = "http://en.wikipedia.org/w/index.php"
params = list(    network=list(    method="append"))
values = c(network='aaaa-bb-cc-dddddd', bla='This is not used!')
ndexr:::ndex_helper_encodeParams(url, params=params, values)
## "http://en.wikipedia.org/w/index.php/aaaa-bb-cc-dddddd"

values = c(bla='This is not used!')
params = list(    network=list(    method="append", optional=TRUE))
ndexr:::ndex_helper_encodeParams(url, params=params, values)
## "http://en.wikipedia.org/w/index.php"

params = list(    network=list(    method="append", default="xxxx-xx-xx-xxxxxx"))
ndexr:::ndex_helper_encodeParams(url, params=params, values)
## "http://en.wikipedia.org/w/index.php/xxxx-xx-xx-xxxxxx"

values = c(network='aaaa-bb-cc-dddddd', bla='This is not used!')
ndexr:::ndex_helper_encodeParams(url, params=params, values)
## "http://en.wikipedia.org/w/index.php/aaaa-bb-cc-dddddd"
```

`ndex_helper_getApi` *Get the Api configuration for a function*

Description

This function extracts the function definition from the ndex configuration within a ndex-connection object. It follows the given path down the list.

Usage

```
ndex_helper_getApi(ndexcon, apiPath)
```

Arguments

`ndexcon`	object of class NDExConnection `ndex_connect`
`apiPath`	character; path to follow in the nested list

Value

configuration of the function

Note

This function is internal.

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Get the function definition for ndex_network_get_summary
## ndex_config[[ndex_config$defaultVersion]]$api$network$summary$get
ndexr:::ndex_helper_getApi(ndexcon, 'network$summary$get')
```

---

ndex_helper_httpResponseHandler

Handles the http server response

Description

This function handles the response from a server. If some response code different from success (200) is returned, the execution stops and the reason is shown.

Usage

```
ndex_helper_httpResponseHandler(response, description, verbose = FALSE)
```

Arguments

response	object of class response (httr)
description	character; description of the action performed
verbose	logical; whether to print out extended feedback

Value

returns the given respons, if it doesn't contain any HTTP error

Note

This function is internal.

Examples

```
ndexr:::ndex_helper_httpResponseHandler(httr::GET('http://www.ndexbio.org'),
                                        'Tried to connect to NDEx server', TRUE)
```

ndex_network_aspect_get_metadata
Get the Metadata Associated with a Network UUID

Description

This function retrieves the metadata associated with the supplied network UUID.

Usage

```
ndex_network_aspect_get_metadata(ndexcon, networkId, aspect)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
networkId	character; unique ID (UUID) of the network
aspect	character; aspect name

Value

metadata for an aspect as list: consistencyGroup, elementCount, lastUpdate, data, name, properties, version and idCounter

REST query

GET: ndex_configapinetwork$aspect$getMetaDataByName

Note

Compatible to NDEx server version 2.0

Server error (version 2.0) since March 13th 2017

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find a network and get its UUID
networks = ndex_find_networks(ndexcon,"p53", "nci-pid")
networkId = networks[1,"externalId"]
## Get the meta-data of an aspect of a network
ndex_network_aspect_get_metadata(ndexcon, networkId, 'nodeAttributes')
```

ndex_network_delete_permission
Delete Network Permission

Description

Removes any permission for the network for the user or group specified

Usage

```
ndex_network_delete_permission(ndexcon, networkId, user = NULL, group = NULL)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
networkId	unique ID of the network
user	character (optional); uuid of the user. Only either user or group may be set!
group	character (optional); uuid of the group. Only either user or group may be set!

Value

1 integer on success, 0 if user/group allready has no permissions on the network

REST query

GET: ndex_configapinetwork$permission$delete

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 1.3 and 2.0

In version 1.3 the function only works for user permissions!

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find one of your networks and get its UUID
# networks = ndex_find_networks(ndexcon, accountName='MyAccountName')
# networkId = networks[1,"externalId"]
## Get the UUID for a user and group
# someUserUuid = "uuuuuuuu-ssss-eeee-rrrr-111111111111"
# someGroupUuid = "ggggggg-rrrr-oooo-uuuu-pppppppppppp"
## Delete the permissions
#ndex_network_delete_permission(ndexcon, networkId, user=someUserUuid)
# => returns 1
#ndex_network_delete_permission(ndexcon, networkId, user=someUserUuid)
```

```
# => returns 0, because user already lost permission on network
#ndex_network_delete_permission(ndexcon, networkId, group=someGroupUuid)
NULL
```

`ndex_network_get_aspect` *Get a Network Aspect As CX*

Description

This function retrieves the provided aspect as CX. The result is the same as accessing an aspect of a RCX object.

Usage

```
ndex_network_get_aspect(ndexcon, networkId, aspect, size)
```

Arguments

`ndexcon`	object of class NDExConnection linkndex_connect
`networkId`	character; unique ID of the network
`aspect`	character; name of the aspect
`size`	integer; specifies the number of elements returned

Details

Note: In future 'ndexr' uses the RCX-object from the corresponding package to handle the networks!

Value

data.frame of the aspect data (the same as rcx[[aspectName]])

REST query

GET: ndex_configapinetwork$aspect$getMetaDataByName

Note

Compatible to NDEx server version 1.3 and 2.0, but doesn't work for version 1.3

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find a network and get its UUID
networks = ndex_find_networks(ndexcon,"p53", "nci-pid")
networkId = networks[1,"externalId"]
## Get the aspect of a network
aspect = ndex_network_get_aspect(ndexcon, networkId, 'nodeAttributes')
# limit the returned elements of the aspect to the first 10 elements
aspect = ndex_network_get_aspect(ndexcon, networkId, 'nodeAttributes', 10)
```

`ndex_network_get_metadata`

Get Network CX Metadata Collection

Description

This function retrieves the (aspect) meta-data of the network identified by the supplied network UUID string.

Usage

```
ndex_network_get_metadata(ndexcon, networkId)
```

Arguments

`ndexcon`	object of class NDExConnection linkndex_connect
`networkId`	character; unique ID (UUID) of the network

Details

Note: In future 'ndexr' uses the RCX-object from the corresponding package to handle the networks! See also Meta-data for more information.

Value

metadata as list: consistencyGroup, elementCount, lastUpdate, name, properties, version and idCounter

REST query

GET: ndex_configapinetwork$aspect$getMetaData

Note

Compatible to NDEx server version 1.3 and 2.0, but doesn't work for version 1.3

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find a network and get its UUID
networks = ndex_find_networks(ndexcon,"p53", "nci-pid")
networkId = networks[1,"externalId"]
## Get the network meta-data
ndex_network_get_metadata(ndexcon, networkId)
```

`ndex_network_get_permission`

Get All Permissions on a Network

Description

This function retrieves the user or group permissions for a network

Usage

```
ndex_network_get_permission(ndexcon, networkId, type, permission, start, size)
```

Arguments

`ndexcon`	object of class NDExConnection linkndex_connect
`networkId`	unique ID of the network
`type`	character ("user"|"group"); specifies whether user or group permissions should be returned
`permission`	character (optional)("READ"|"WRITE"|"ADMIN"); constrains the type of the returned membership. If not set (or NULL), all permission types will be returned.
`start`	integer (optional); specifies that the result is the nth page of the requested data.
`size`	integer (optional); specifies the number of data items in each page.

Value

data.frame containing user or group UUIDs and the highest permission assigned to that user or group

REST query

GET: ndex_configapinetwork$permission$get

Note

Compatible to NDEx server version 1.3 and 2.0

In version 1.3 the function only returns user permissions and differs in the returned data (more columns)!

Requires an authorized user! (ndex_connect with credentials)

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find one of your networks and get its UUID
# networks = ndex_find_networks(ndexcon, accountName='MyAccountName')
# networkId = networks[1,"externalId"]
## Get the permissions
# permissions = ndex_network_get_permission(ndexcon, networkId, 'user')
## Version 2.0:
## names(permission)
## [1] "memberUUID" "permission"
## Version 1.3:
## names(permission)
## [1] "membershipType"    "memberUUID"  "resourceUUID"
## [4] "memberAccountName" "permissions" "resourceName"
# permissions = ndex_network_get_permission(ndexcon, networkId, 'user', NULL)   # same as previous
# permissions = ndex_network_get_permission(ndexcon, networkId, 'user', 'READ', 0, 10)
# permissions = ndex_network_get_permission(ndexcon, networkId, 'group')
NULL
```

---

ndex_network_get_provenance

Get Network Provenance

Description

This function retrieves the provenance of the network identified by the supplied network UUID string.

Usage

```
ndex_network_get_provenance(ndexcon, networkId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
networkId	unique ID of the network

Value

List of network metadata: ID, name, whether it is public, edge and node count; source and format of network

REST query

GET: ndex_configapinetwork$provenance$get

Note

Compatible to NDEx server version 1.3 and 2.0

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find a network and get its UUID
networks = ndex_find_networks(ndexcon,"p53", "nci-pid")
networkId = networks[1,"externalId"]
## Get the network provenace
provenance = ndex_network_get_provenance(ndexcon, networkId)
```

---

ndex_network_get_summary

Get NetworkSummary by Network UUID

Description

This function retrieves the summary of the network identified by the supplied network UUID string.

Usage

```
ndex_network_get_summary(ndexcon, networkId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
networkId	unique ID of the network

Value

List of network metadata: ID, name, whether it is public, edge and node count; source and format of network

REST query

GET: ndex_configapinetwork$summary$get

Note

Compatible to NDEx server version 1.3 and 2.0

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Find a network and get its UUID
networks = ndex_find_networks(ndexcon,"p53", "nci-pid")
networkId = networks[1,"externalId"]
## Get the network summary
summary = ndex_network_get_summary(ndexcon, networkId)
```

---

ndex_network_set_systemProperties

Set Network System Properties

Description

Network System properties are the properties that describe the network's status in a particular NDEx server but that are not part of the corresponding CX network object.

Usage

```
ndex_network_set_systemProperties(
  ndexcon,
  networkId,
  readOnly = NULL,
  visibility = NULL,
  showcase = NULL
)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
networkId	unique ID of the network
readOnly	boolean (optional); Sets the network to only readable. At least one of readOnly, visibility or showcase have to be set!
visibility	character (optional) ('PUBLIC'|'PRIVATE'); Sets the network to only readable. At least one of readOnly, visibility or showcase have to be set!
showcase	boolean (optional); Authenticated user can use this property to control whether this network will display in his or her home page. Caller will receive an error if the user does not have explicit permission to that network. At least one of readOnly, visibility or showcase have to be set!

Value

NULL on success; Error else

REST query

GET: ndex_configapinetwork$systemproperties$set

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 1.3 and 2.0

In version 1.3 only the parameter readOnly is supported

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find one of your networks and get its UUID
# networks = ndex_find_networks(ndexcon, accountName='MyAccountName')
# networkId = networks[1,"externalId"]
## Set network system properties
# ndex_network_set_systemProperties(ndexcon, networkId, readOnly=TRUE)
# ndex_network_set_systemProperties(ndexcon, networkId, visibility="PUBLIC")
# ndex_network_set_systemProperties(ndexcon, networkId, showcase=TRUE)
# ndex_network_set_systemProperties(ndexcon, networkId,
#                                   readOnly=FALSE, visibility="PRIVATE", showcase=FALSE)
NULL
```

---

ndex_network_update_aspect

Update an Aspect of a Network

Description

This function updates an aspect with the provided CX for the aspect.

Usage

```
ndex_network_update_aspect(
  ndexcon,
  networkId,
  aspectName,
  aspectAsRCX,
  isJson = FALSE
)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
networkId	unique ID of the network
aspectName	name of the aspect
aspectAsRCX	rcx data for the aspect (rcx[[aspectName]])
isJson	logical if aspectAsRCX is already JSON

Value

networkId unique ID of the modified network

REST query

PUT: ndex_configapinetwork$aspect$update

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find one of your networks and get its UUID
# networks = ndex_find_networks(ndexcon, accountName='MyAccountName')
# networkId = networks[1,"externalId"]
## Get the network data
# aspect = ndex_network_get_aspect(ndexcon, networkId, 'nodeAttributes')
## Do some changes to the aspect..
# aspectModified = aspect[1:5,]
## and update the aspect
# ndex_network_update_aspect(ndexcon,pws[1,"externalId"], 'nodeAttributes', aspectModified)
NULL
```

---

ndex_network_update_permission

Update Network Permission

Description

Updates the permission of a user specified by userid or group specified by groupid for the network

Usage

```
ndex_network_update_permission(
  ndexcon,
  networkId,
  user = NULL,
  group = NULL,
  permission
)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
networkId	unique ID of the network
user	character (optional); uuid of the user. Only either user or group may be set!
group	character (optional); uuid of the group. Only either user or group may be set!
permission	character (optional)("READ"|"WRITE"|"ADMIN"); type of permission to be given. If granted admin permission, the current admin loses the admin status.

Value

1 integer on success, 0 if user/group allready has this permissions on the network

REST query

GET: ndex_configapinetwork$permission$update

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 1.3 and 2.0, but doesn't work for version 1.3

In version 1.3 the function only works for user permissions!

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find one of your networks and get its UUID
# networks = ndex_find_networks(ndexcon, accountName='MyAccountName')
# networkId = networks[1,"externalId"]
## Get the UUID for a user and group
# someUserUuid = "uuuuuuuu-ssss-eeee-rrrr-111111111111"
# someGroupUuid = "ggggggg-rrrr-oooo-uuuu-pppppppppppp"
## Change the permissions
# ndex_network_update_permission(ndexcon, networkId, user=someUserUuid, 'WRITE')
# ndex_network_update_permission(ndexcon, networkId, group=someGroupUuid, 'READ')
## Set a new admin (lose own admin status)
# ndex_network_update_permission(ndexcon, networkId, user=someUserUuid, 'ADMIN')
NULL
```

`ndex_network_update_profile` *Update Network Profile*

Description

Updates the profile information of the network. Any profile attributes specified will be updated but attributes that are not specified will have no effect - omission of an attribute does not mean deletion of that attribute.

Usage

```
ndex_network_update_profile(
  ndexcon,
  networkId,
  name = NULL,
  description = NULL,
  version = NULL
)
```

Arguments

`ndexcon`	object of class NDExConnection linkndex_connect
`networkId`	unique ID of the network
`name`	character (optional); Changes the name the network. At least one of name, description or version have to be set!
`description`	character (optional); Changes the description the network. At least one of name, description or version have to be set!
`version`	character (optional); Changes the version the network. At least one of name, description or version have to be set!

Value

NULL on success; Error else

REST query

GET: ndex_configapinetwork$profile$update

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 1.3 and 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find one of your networks and get its UUID
# networks = ndex_find_networks(ndexcon, accountName='MyAccountName')
# networkId = networks[1,"externalId"]
## Update network profile
# ndex_network_update_profile(ndexcon, networkId, name="Some fancy name for the network")
# ndex_network_update_profile(ndexcon, networkId, description="Description of the network")
# ndex_network_update_profile(ndexcon, networkId, version="1.2.3.4")
# ndex_network_update_profile(ndexcon, networkId, name="Special test network",
#                             description="Nothing to see here", version="1.3")
NULL
```

---

ndex_rest_DELETE *Generic DELETE query to API.*

Description

Generic DELETE query to API.

Usage

```
ndex_rest_DELETE(ndexcon, route, raw = FALSE)
```

Arguments

ndexcon	object of class NDExConnection ndex_connect
route	Character (route to specific REST query)
raw	Specifies if server response should be returned in raw, or if jsonlite::fromJSON is called first. Defaults to FALSE.

Details

Simply execute HTTP DELETE on URL host/route and fetch whatever data REST server returns Making sure the route is well-formed is the job of calling function

Value

JSON response from REST server, NULL if no valid JSON was received. if parameter raw is TRUE, the raw response is returned without a call to jsonlite::fromJSON.

Note

This function is internal.

See Also

ndex_rest_GET, ndex_rest_POST, ndex_rest_PUT and ndex_rest_DELETE

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Not run:
ndex_rest_DELETE(ndexcon, "/networks/api")

## End(Not run)
```

---

ndex_rest_GET *Generic GET query to API.*

Description

Generic GET query to API.

Usage

```
ndex_rest_GET(ndexcon, route, raw = FALSE)
```

Arguments

ndexcon	object of class NDExConnection ndex_connect
route	Character (route to specific REST query)
raw	Specifies if server response should be returned in raw, or if jsonlite::fromJSON is called first. Defaults to FALSE.

Details

Simply execute HTTP GET on URL host/route and fetch whatever data REST server returns Making sure the route is well-formed is the job of calling function

Value

JSON response from REST server, NULL if no valid JSON was received. if parameter raw is TRUE, the raw response is returned without a call to jsonlite::fromJSON.

Note

This function is internal.

See Also

ndex_rest_GET, ndex_rest_POST, ndex_rest_PUT and ndex_rest_DELETE

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Not run:
ndex_rest_GET(ndexcon, "/networks/api")

## End(Not run)
```

---

ndex_rest_POST *Generic POST query to API*

Description

Generic POST query to API

Usage

```
ndex_rest_POST(ndexcon, route, data, multipart = FALSE, raw = FALSE)
```

Arguments

ndexcon	object of class NDExConnection ndex_connect
route	Character (route to specific REST query)
data	Whatever data to be supplied with query. Should be valid JSON
multipart	Whatever data to be supplied with query. Should be valid JSON
raw	Specifies if server response should be returned in raw, or if jsonlite::fromJSON is called first. Defaults to FALSE.

Details

Simply execute HTTP POST on URL host/route and fetch whatever data REST server returns Making sure the route is well-formed is the job of calling function Making sure the data is well-formed is also the job of calling function

Value

JSON response from REST server, NULL if no valid JSON was received. if parameter raw is TRUE, the raw response is returned without a call to jsonlite::fromJSON.

Note

This function is internal.

See Also

ndex_rest_GET, ndex_rest_POST, ndex_rest_PUT and ndex_rest_DELETE

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Not run:
ndex_rest_POST(ndexcon, "/networks/api", data)
ndex_rest_POST(ndexcon, "/networks/api", data, raw=TRUE)
ndex_rest_POST(ndexcon, "/networks/api", list(some=data, other=data2), multipart=TRUE)

## End(Not run)
```

---

ndex_rest_PUT *Generic PUT query to API*

Description

Generic PUT query to API

Usage

```
ndex_rest_PUT(ndexcon, route, data = NULL, multipart = FALSE, raw = FALSE)
```

Arguments

ndexcon	object of class NDExConnection ndex_connect
route	Character (route to specific REST query)
data	Whatever data to be supplied with query. Should be valid JSON
multipart	Whatever data to be supplied with query. Should be valid JSON
raw	Specifies if server response should be returned in raw, or if jsonlite::fromJSON is called first. Defaults to FALSE.

Details

Simply execute HTTP PUT on URL host/route and fetch whatever data REST server returns Making sure the route is well-formed is the job of calling function Making sure the data is well-formed is also the job of calling function

Value

JSON response from REST server, NULL if no valid JSON was received. if parameter raw is TRUE, the raw response is returned without a call to jsonlite::fromJSON.

Note

This function is internal.

See Also

ndex_rest_GET, ndex_rest_POST, ndex_rest_PUT and ndex_rest_DELETE

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## Not run:
ndex_rest_PUT(ndexcon, "/networks/api", data)
ndex_rest_PUT(ndexcon, "/networks/api", data, raw=TRUE)
ndex_rest_PUT(ndexcon, "/networks/api", list(some=data, other=data2), multipart=TRUE)

## End(Not run)
```

---

ndex_update_group *Update Group*

Description

Updates the group based on the data.

Usage

```
ndex_update_group(
  ndexcon,
  groupId,
  groupName,
  image,
  website,
  description,
  properties
)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
groupId	character; unique ID (UUID) of the group
groupName	character; name of the new graoup
image	character (optional); URL of the account owner's image.
website	character (optional); URL of the account owner's web site
description	character (optional); Short description of this user.
properties	list (optional); additional properties for the group

Value

Empty string ("") on success, else error

REST query

PUT: ndex_configapiuser$update

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find user and get its id
# user = ndex_find_user_byName(ndexcon, 'MyAccountName')
# userId = user$externalId
## Find the user's groups and get one group id
# groups = ndex_user_list_groups(ndexcon, userId)
# groupId = groups[1,"externalId"]
## Update the group
# ndex_update_group(ndexcon, groupId, description='A really nice group!')
NULL
```

---

ndex_update_network *Update an Entire Network as CX*

Description

Note: In future 'ndexr' uses the RCX-object from the corresponding package to handle the networks!

Usage

```
ndex_update_network(ndexcon, rcx, networkId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
rcx	RCX-object object
networkId	(optional); unique ID of the network

Details

This method updates/replaces a existing network on the NDEx server with new content from the given RCX object. The UUID can either be specified manually or it will be extracted from the RCX object (i.e. from rcx$ndexStatus$externalId).

Value

UUID of the updated network

REST query

PUT (multipart/form-data): ndex_configapinetwork$update$url data: CXNetworkStream = data

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 1.3 and 2.0

Examples

```
## Establish a server connections with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find one of your networks and get its UUID
# networks = ndex_find_networks(ndexcon, accountName='MyAccountName')
# networkId = networks[1,"externalId"]
## Get the network data
# rcx = ndex_get_network(ndexcon, networkId)
## Do some changes to rcx..
## and update the network
# networkId = ndex_update_network(ndexcon, rcx, networkId)
# networkId = ndex_update_network(ndexcon, rcx) ## same as previous
NULL
```

---

ndex_update_user *Update User*

Description

Updates the authenticated user based on the data. Errors, if the user for ndexcon and uuid are different.

Usage

```
ndex_update_user(
  ndexcon,
  userId,
  emailAddress,
  isIndividual,
  displayName,
  firstName,
  lastName,
  image,
  website,
  description,
  verbose = FALSE
)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userId	character; unique ID of the user
emailAddress	character (optional); email address (used for verification if enabled)
isIndividual	boolean (default:True); True if this account is for an individual user. False means this account is for an organization or a project etc.
displayName	character (optional); Display name of this account, only applied to non-individual accounts.
firstName	character (optional); Account owner's first name, only applies to individual accounts.
lastName	character (optional); Account owner's last name, only appliies to individual accounts.
image	character (optional); URL of the account owner's image.
website	character (optional); URL of the account owner's web site
description	character (optional); Short description of this user.
verbose	logical (optional); whether to print out extended feedback

Value

Empty string ("") on success, else error

REST query

GET: ndex_configapiuser$update

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find user and get its id
# user = ndex_find_user_byName(ndexcon, 'SomeUserName')
# userId = user$externalId
## Update user
# ndex_update_user(ndexcon, userId, firstName = 'Homer Jay', lastName = 'Simpson')
# ndex_update_user(ndexcon, userId, displayName = 'Max Power',
#            image='https://upload.wikimedia.org/wikipedia/en/0/02/Homer_Simpson_2006.png',
#               description='One of the most influential characters in the history of TV')
NULL
```

ndex_user_change_password *Change Password*

Description

Changes the authenticated user's password to the new password

Usage

```
ndex_user_change_password(ndexcon, userId, password)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userId	character; unique ID of the user
password	character; New password

Value

Empty string on success, else error

REST query

GET: ndex_configapiuser$password$change

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find user and get its id
# user = ndex_find_user_byName(ndexcon, 'SomeUserName')
# userId = user$externalId
## Change user password
# ndex_user_change_password(ndexcon, userId, 'SuperSaveNewPassword')
NULL
```

ndex_user_forgot_password *Forgot Password*

Description

Causes a new password to be generated for the given user account and then emailed to the user's emailAddress

Usage

```
ndex_user_forgot_password(ndexcon, userId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userId	character; unique ID of the user

Value

Empty string on success, else error

REST query

GET: ndex_configapiuser$password$mail Wrapper for ndex_user_mail_password()

Note

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection
# ndexcon = ndex_connect()
## Find user and get its id
# user = ndex_find_user_byName(ndexcon, 'SomeUserName')
# userId = user$externalId
## Request new password via email
# ndex_user_forgot_password(ndexcon, userId)
NULL
```

ndex_user_get_networksummary
Get User's Account Page Networks

Description

This is a convenience function designed to support "My Account" pages in NDEx applications. It returns a list of NetworkSummary objects to display.

Usage

```
ndex_user_get_networksummary(ndexcon, userId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userId	character; unique ID (UUID) of the user

Value

data.frame of networks (name, description, externalId, uri, etc.) on the account page of the specified user

REST query

GET: ndex_configapiuser$networksummary

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## get user by name to get UUID
# user = ndex_find_user_byName(ndexcon, 'MyAccountName')
# userId = user$externalId
## get all network permissions of the user
# networkSummary = ndex_user_get_networksummary(con, user$externalId)
# names(networkSummary)
## [1] "ownerUUID"    "isReadOnly"  "subnetworkIds"  "errorMessage"      "isValid"
## [6] "warnings"     "isShowcase"  "visibility"     "edgeCount"         "nodeCount"
##[11] "uri"          "version"     "owner"          "name"              "properties"
##[16] "description"  "externalId"  "isDeleted"      "modificationTime"  "creationTime"
NULL
```

ndex_user_get_showcase *Get User's Showcase Networks*

Description

This is a convenience function to support "user pages" in NDEx applications. This function returns a list of network summary objects that the user who is specified by userid chose to display in his or her home page. For authenticated users, this function returns the networks that the authenticated user can read, for anonymous users, this function returns only public networks.

Usage

```
ndex_user_get_showcase(ndexcon, userId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userId	character; unique ID (UUID) of the user

Value

data.frame of networks (name, description, externalId, uri, etc.) in the showcase of the specified user

REST query

GET: ndex_configapiuser$showcase

Note

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection
ndexcon = ndex_connect()
## get user by name to get UUID
user = ndex_find_user_byName(ndexcon, 'ndextutorials')
userId = user$externalId
## get all network permissions of the user
showcase = ndex_user_get_showcase(ndexcon, userId)
names(showcase)
## [1] "ownerUUID"    "isReadOnly"  "subnetworkIds"  "errorMessage"      "isValid"
## [6] "warnings"     "isShowcase"  "visibility"     "edgeCount"         "nodeCount"
##[11] "uri"          "version"     "owner"          "name"              "properties"
##[16] "description"  "externalId"  "isDeleted"      "modificationTime"  "creationTime"
```

ndex_user_list_groups *Get User's Group Memberships*

Description

Query finds groups for which the current user has the specified membership type. If the "type' parameter is omitted, all membership types will be returned. Returns a map which maps a group UUID to the membership type the authenticated user has.

Usage

```
ndex_user_list_groups(ndexcon, userId, type = NULL, start = NULL, size = NULL)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userId	character; unique ID (UUID) of the user
type	character (optional)("MEMBER"|"GROUPADMIN"); constrains the type of the returned membership. If not set (or NULL), all permission types will be returned.
start	integer (optional); specifies that the result is the nth page of the requested data.
size	integer (optional); specifies the number of data items in each page.

Value

List of permissions of that user or empty object

REST query

GET: ndex_configapiuser$group$list

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find user and get its id
# user = ndex_find_user_byName(ndexcon, 'MyAccountName')
# userId = user$externalId
## Find the user's groups and get one group id
# groups = ndex_user_list_groups(ndexcon, userId)
## $`ggggggg-rrrr-oooo-uuuu-pppppp111111`
## [1] "MEMBER"
```

```
##
## $`ggggggg-rrrr-oooo-uuuu-pppppp222222`
## [1] "GROUPADMIN"
# groupIds = names(groups)
## [1] "ggggggg-rrrr-oooo-uuuu-pppppp111111" "ggggggg-rrrr-oooo-uuuu-pppppp222222"
NULL
```

`ndex_user_list_permissions` *Get User's Network Permissions*

Description

This function returns networks for which the authenticated user is assigned the specified permission. Userid is the UUID of the authenticated user. Returns a JSON map in which the keys are network UUIDs and values are the highest permission assigned to the authenticated user.#'

Usage

```
ndex_user_list_permissions(
  ndexcon,
  userId,
  type = NULL,
  directonly = FALSE,
  start = NULL,
  size = NULL
)
```

Arguments

`ndexcon`	object of class NDExConnection linkndex_connect
`userId`	character; unique ID (UUID) of the user
`type`	character (optional)("READ"|"WRITE"|"ADMIN"); constrains the type of the returned permission. If not set (or NULL), all permission types will be returned.
`directonly`	logical (default: FALSE); If directonly is set to true, permissions granted through groups are not included in the result
`start`	integer (optional); specifies that the result is the nth page of the requested data.
`size`	integer (optional); specifies the number of data items in each page.

Value

List of highest permissions of that user or empty object

REST query

GET: ndex_configapiuser$permission$list

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## get user by name to get UUID
# user = ndex_find_user_byName(ndexcon, 'MyAccountName')
# userId = user$externalId
## get all network permissions of the user
# networkPermissions = ndex_user_list_permissions(ndexcon, userId)
## $`nnneeett-wwww-oooo-rrrr-kkkkkkk11111`
## [1] "ADMIN"
## $`nnneeett-wwww-oooo-rrrr-kkkkkkk22222`
## [1] "WRITE"
## $`nnneeett-wwww-oooo-rrrr-kkkkkkk33333`
## [1] "READ"
# networkIds = names(networkPermissions)
## [1] "nnneeett-wwww-oooo-rrrr-kkkkkkk11111" "nnneeett-wwww-oooo-rrrr-kkkkkkk22222"
## [3] "nnneeett-wwww-oooo-rrrr-kkkkkkk33333"
## get all networks for which the user has Admin permissions
# networkPermissions = ndex_user_list_permissions(ndexcon, userId, type='ADMIN')
## $`nnneeett-wwww-oooo-rrrr-kkkkkkk11111`
## [1] "ADMIN"
## get all networks for which the user has direct access
# networkPermissions = ndex_user_list_permissions(ndexcon, user$externalId, directonly=TRUE)
## $`nnneeett-wwww-oooo-rrrr-kkkkkkk11111`
## [1] "ADMIN"
NULL
```

---

ndex_user_mail_password

Email New Password

Description

Causes a new password to be generated for the given user account and then emailed to the user's emailAddress

Usage

```
ndex_user_mail_password(ndexcon, userId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userId	character; unique ID of the user

Value

Empty string on success, else error

REST query

GET: ndex_configapiuser$password$mail

Note

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection
# ndexcon = ndex_connect()
## Find user and get its id
# user = ndex_find_user_byName(ndexcon, 'SomeUserName')
# userId = user$externalId
## Request new password via email
# ndex_user_mail_password(ndexcon, userId)
NULL
```

---

ndex_user_show_group *Get User's Membership in Group*

Description

Returns the permission that the user specified in the URL has on the given group. Returns an empty object if the authenticated user is not a member of this group.

Usage

```
ndex_user_show_group(ndexcon, userId, groupId)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userId	character; unique ID (UUID) of the user
groupId	character; unique ID (UUID) of the group

Value

List of permissions of that user or empty object

REST query

GET: ndex_configapiuser$group$get

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## Find user and get its id
# user = ndex_find_user_byName(ndexcon, 'SomeUserName')
# userId = user$externalId
## Find the user's groups and get one group id
# groups = ndex_user_list_groups(ndexcon, userId)
# groupId = groups[1,"externalId"]
## get users's permission in the group
# userPermissions = ndex_user_show_group(ndexcon, userId, groupId)
## $`uuuuuuuu-ssss-eeee-rrrr-123456789abc`
## [1] "MEMBER"
NULL
```

---

ndex_user_show_permission

Get User's Permission for Network

Description

Get the type(s) of permission assigned to the authenticated user for the specified network. Returns a map which maps a network UUID to the highest permission assigned to the authenticated user.

Usage

```
ndex_user_show_permission(ndexcon, userId, networkId, directonly = FALSE)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userId	character; unique ID (UUID) of the user
networkId	character; unique ID (UUID) of the group
directonly	logical (default: FALSE); If directonly is set to true, permissions granted through groups are not included in the result

Value

List of permissions of that user ("READ"|"WRITE"|"ADMIN") or empty object

REST query

GET: ndex_configapiuser$permission$get

Note

Requires an authorized user! (ndex_connect with credentials)

Compatible to NDEx server version 2.0

See Also

ndex_network_get_permission

Examples

```
## Establish a server connection with credentials
# ndexcon = ndex_connect('MyAccountName', 'MyPassword')
## get user by name to get UUID
# user = ndex_find_user_byName(ndexcon, 'MyAccountName')
# userId = user$externalId
## Find one of your networks and get its UUID
# networks = ndex_find_networks(ndexcon, accountName='MyAccountName')
# networkId = networks[1,"externalId"]
## get users's permission to a network
# networkPermissions = ndex_user_show_permission(ndexcon, userId, networkId, directonly=TRUE)
## $`nnneeett-wwww-oooo-rrrr-kkkkkkk11111`
## [1] "ADMIN"
NULL
```

---

ndex_verify_user *Verify a User*

Description

Verify the given user with UUID and verification code, which is set by email

Usage

```
ndex_verify_user(ndexcon, userId, code)
```

Arguments

ndexcon	object of class NDExConnection linkndex_connect
userId	character; unique ID of the user
code	character; Verification code sent by email

Value

string "User account XXX has been activated." when this user's account is successfully activated.

REST query

GET: ndex_configapiuser$verify

Note

Compatible to NDEx server version 2.0

Examples

```
## Establish a server connection
# ndexcon = ndex_connect()
## Find user and get its id
# user = ndex_find_user_byName(ndexcon, 'SomeUserName')
# userId = user$externalId
## Verify user with verification code
# ndex_verify_user(ndexcon, userId, 'Osqy11mRZ9')
## [1] "User account XXX has been activated."
NULL
```

---

print.NDExConnection *Print a NDExConnection object*

Description

This function creates an NDExConnection which stores options and authentication details. It is a parameter required for most of the other ndexr functions. If username and password are missing an anonymous connection is created, which already offers most of the retrieval functionality.

Usage

```
## S3 method for class 'NDExConnection'
print(x, ...)
```

Arguments

x	NDExConnection; stores options, authentication and api configuration
...	further arguments passed to or from other methods.

Value

Just prints the NDExConnection object

See Also

ndex_connect and ndex_config

Examples

```
ndexcon = ndex_connect()   ## log in anonymously
print(ndexcon)
ndexcon = ndex_connect(verbose=TRUE)   ## same as above with extended feedback
print(ndexcon)
## Not run:
## log in with credentials
ndexcon = ndex_connect('user','password')
print(ndexcon)
## running some NDEx server locally
ndexcon = ndex_connect(host='localhost:8765')
print(ndexcon)
## manually change the api and connection configuration
ndexcon = ndex_connect(ndexConf=ndex_config$Version_2.0)
print(ndexcon)

## End(Not run)
```

Index