Package 'ndexr'

April 12, 2022

Type Package

Title NDEx R client library

Version 1.16.0

Date 2018-04-27

Author
Florian Auer <florian.auer@informatik.uni-augsburg.de>, Frank Kramer <frank.kramer@informatik.uni-augsb
ter Pratt <depratt@ucsc.edu>

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

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

Depends igraph

Imports httr, jsonlite, plyr, tidyr

RoxygenNote 7.1.1

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 RELEASE_3_14

git_last_commit 26abbfb

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

R topics documented:

`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 object and as igraph representation.

Details

Package:	ndexr
Type:	Package
Version:	2.0.7
Date:	2016-12-02
License:	TBD

Author(s)

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

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

Alex Ishkin <aleksandr.ishkin@thomsonreuters.com>

Dexter Pratt <depratt@ucsc.edu>

Examples

```
## Not run:
require(ndexr)
###connect anonymously
ndexcon1 = ndex_connect(verbose=T)
###get network api
apidata1 = ndex_get_network.api(ndexcon1)
###find some networks containing p53
pws1 = ndexr::ndex_find_networks(ndexcon1,"p53")
###get complete network as RCX
```

```
rcx1 = ndex.get.complete.network(ndexcon1,pws1[1,"externalId"])
###convert to rcxgraph and back
rcxgraph1 = rcx_toRCXgraph(rcx1)
plot(rcxgraph1, vertex.label=V(rcxgraph1)$n, edge.label=E(rcxgraph1)$i)

## End(Not run)
```

`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_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

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 CX. 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

Usage

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

Arguments

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

Value

RCX 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_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 = "gggggggg-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

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

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)
```

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]])

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_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

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).

Usage

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

Arguments

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

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
```

rcxgraph_toRCX *Create RCX object from RCXgraph object*

Description

This function creates an RCX object from a valid RCXgraph object.

The following rules apply to convert from RCXgraph to RCX:

- all graph attributes are stored as named data.frames within the RCX object
- nodes receive their name value as "@id" attribute. All other node attributes are saved in the RCX object as nodeAttributes, access via rcx[["nodeAttributes"]]. Data goes from wide format (columns for each unique n with cells contianing v) to long format (column n containing attribute name and column v containing attribute value).
- edges are connected via their "s"art and "t"arget fields. The "@id" and "i"nteraction attribute are stored as is and all edgeAttributes are saved as node attributes, access via rcx[["edgeAttributes"]]. Data goes from wide format (columns for each unique n with cells contianing v) to long format (column n containing attribute name and column v containing attribute value).

Usage

```
rcxgraph_toRCX(rcxgraph, verbose = FALSE)

rcx_fromRCXgraph(rcxgraph, verbose = FALSE)
```

Arguments

rcxgraph	RCX object
verbose	logical; whether to print out extended feedback

Details

An RCXgraph object could look like this:

```
## Get some network...
> ndexcon = ndex_connect()
> rcx = ndex_get_network(ndexcon,"dd268e2f-fd4d-11e7-adc1-0ac135e8bacf")

> summary(rcxgraph)
IGRAPH f99ed1e DN-- 30 218 --
+ attr: metaData (g/x), numberVerification (g/x), ndexStatus (g/x), provenanceHistory (g/x), networkAtt
| (g/x), cyHiddenAttributes (g/x), status (g/x), name (v/c), @id (v/n), n (v/c), Basal (v/c), avg_PPR (v
| (v/c), Her2 (v/c), name (e/c), @id (e/n), i (e/c), NAME (e/c), strength (e/c), interaction (e/c)

## Attributes marked with "(g/x)" are graph attributes, "(v/x)" and "(e/x)" correspond to vertex and edg
## Accessing a graph attribute, e.g. "metaData"
rcxgraph$metaData
```

```
  consistencyGroup elementCount  lastUpdate          name properties version idCounter
1              1             1 1.516391e+12       ndexStatus      NULL    1.0       NA
2              1             1 1.516391e+12  provenanceHistory      NULL    1.0       NA
3              1            30          NA            nodes      NULL    1.0      826
4              1           218          NA            edges      NULL    1.0      827
5              1            11          NA networkAttributes      NULL    1.0       NA
6              1           210          NA    nodeAttributes      NULL    1.0       NA
7              1           654          NA    edgeAttributes      NULL    1.0       NA
8              1            30          NA   cartesianLayout      NULL    1.0       NA
9              1             3          NA cyVisualProperties      NULL    1.0       NA
10             1             1          NA cyHiddenAttributes      NULL    1.0       NA


## The vertices:
> V(rcxgraph)[[]]
+ 30/30 vertices, named, from f99ed1e:
   name X.id    n        Basal       avg_PPR         LumA         LumB      ANOVA_FDR        ANOVA_p
1   ABL1  332  ABL1 2.10282304283E-4 0.00153239525422 2.40855043132E-4 0.00243281556888  0.0126090186
2   JAK2  331  JAK2 0.00818834880039 0.00229451204468 0.00112294137386 5.02052428871E-4  2.1649538801
3 NOTCH4  330 NOTCH4 0.00271376234049 0.00146318626692 9.53762145731E-4 0.00122487574678  2.1649538801
4  KMT2C  329  KMT2C 0.00628341277798 0.00228045774193 0.00156001096343 0.00150488341005  0.0227077886
5 NOTCH1  328 NOTCH1 0.00839759047274  0.0185377425722  0.0238783637494  0.0174878270867 5.65380895701E
6 ...

## Display the igraph vertex names
> V(rcxgraph)$name
 [1] "ABL1"   "JAK2"   "NOTCH4" "KMT2C"  "NOTCH1" "INPPL1" "SPOP"   "AKT1"   "MYC"    "GATA3"  "MET"    "PTPR
[16] "IBSP"   "ERBB2"  "PIK3CA" "EGFR"   "TLN1"   "MYCN"   "KRAS"   "CDH1"   "TP53"   "CCNE1"  "CCND1"  "CARD

## The igraph vertex names equal to the ndex node names by default
> V(rcxgraph)$n
 [1] "ABL1"   "JAK2"   "NOTCH4" "KMT2C"  "NOTCH1" "INPPL1" "SPOP"   "AKT1"   "MYC"    "GATA3"  "MET"    "PTPR
[16] "IBSP"   "ERBB2"  "PIK3CA" "EGFR"   "TLN1"   "MYCN"   "KRAS"   "CDH1"   "TP53"   "CCNE1"  "CCND1"  "CARD

## If in \link{rcx_toRCXgraph} or \link{rcxgraph_fromRCX} \quote{idAsVertexName} is set, the igraph ver
## Note: "@id" is displayed in the vertex summary as "X.id", but can be accessed using "@id"
> V(rcxgraph)$'@id'
[1] 332 331 330 329 328 327 326 325 324 323 322 321 320 319 318 317 316 315 314 313 312 311 310 309 308 307


## The edges:
> E(rcxgraph)[[]]
+ 218/218 edges from f99ed1e (vertex names):
    tail   head tid hid        name X.id          i                    NAME       strength   interaction
1   ABL1   EGFR   1  19 interacts with  550 interacts with    ABL1 (interacts with) EGFR  2.6501871987E-8
2   ABL1  GATA3   1  10 interacts with  549 interacts with   ABL1 (interacts with) GATA3  0.0017942365068
3   ABL1 INPPL1   1   6 interacts with  548 interacts with  ABL1 (interacts with) INPPL1  0.0053112520138
4   ABL1   JAK2   1   2 interacts with  547 interacts with    ABL1 (interacts with) JAK2  0.00668830637406
```

```
5    ABL1   MET  1  11 interacts with  546 interacts with    ABL1 (interacts with) MET  2.37014815511E-5
6 ...

## Display the igraph edge names
> E(rcxgraph)$name
[1] "interacts with" "interacts with" "interacts with" "interacts with" "interacts with" "interacts wit
[8] ...

## The igraph edge names equal to the ndex edge "i"nteraction by default
> E(rcxgraph)$i
[1] "interacts with" "interacts with" "interacts with" "interacts with" "interacts with" "interacts wit
[8] ...

## If in \link{rcx_toRCXgraph} or \link{rcxgraph_fromRCX} \quote{idAsEdgeName} is set, the igraph edge
## Note: "@id" is displayed in the edge summary as "X.id", but can be accessed using "@id"
> E(rcxgraph)$'@id'
 [1] 550 549 548 547 546 545 544 543 542 541 540 539 538 537 536 535 534 533 532 531 530 529 528 527 526 525
[30] ...
```

Value

returns object of class RCX if successfull, NULL otherwise

See Also

rcxgraph_fromRCX rcx_toRCXgraph rcx_fromJSON rcx_toJSON RCX igraph

Examples

```
## Create an RCX object
rcx = rcx_new(c('@id'=1, n='Some Name', r='HGNC:Symbol'))

## Convert to RCXgraph
rcxgraph = rcxgraph_fromRCX(rcx)
## or
rcxgraph = rcx_toRCXgraph(rcx)

## Convert RCXgraph back to RCX
rcx = rcx_fromRCXgraph(rcxgraph)
## or
rcx = rcxgraph_toRCX(rcxgraph)
```

---

rcx_asNewNetwork *Remove all interfering NDEx artefacts from RCX object*

Description

Remove all interfering NDEx artefacts from RCX object

Usage

```
rcx_asNewNetwork(rcx)
```

Arguments

rcx	RCX object

Details

After a RCX object is downloaded from an NDEx server, it will contain some aspects that are not present in a newly generated network, i.e. ndexStatus', provenanceHistory' and 'status'. Removing those aspects might be useful in some cases.

Value

RCX object

See Also

rcx_fromJSON

Examples

```
## Create an RCX object
rcx = rcx_new(c('@id'=1, n='Some Name', r='HGNC:Symbol'))
## Remove NDEx artefacts
rcx = rcx_asNewNetwork(rcx)
## Not run:
rcxjson = rcx_toJSON(rcx)
ndex_create_network(ndexcon, rcxjson)

## End(Not run)
```

---

rcx_fromJSON *Create RCX object from JSON data*

Description

This function creates an RCX object from a supplied JSON-encoded CX object. RCX objects store the CX data as a named list of data.frames containing metaData and all aspects of the network.

Usage

```
rcx_fromJSON(json, verbose = FALSE)
```

Arguments

json	JSON data
verbose	logical; whether to print out extended feedback

Details

The structure of an RCX object, as shown via str(rcx) could be a list like this:

```
> str(rcx)

List of 12
$ metaData          :'data.frame':    11 obs. of  7 variables:
  ..$ name        : chr [1:11] "citations" "@context" "edgeAttributes" "edgeCitations" ...
  ..$ consistencyGroup: int [1:11] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ elementCount    : int [1:11] 4 23 NA NA 11 1 NA NA NA 5 ...
  ..$ lastUpdate     : num [1:11] 1.44e+12 1.44e+12 1.44e+12 1.44e+12 1.44e+12 ...
  ..$ version         : chr [1:11] "1.0" "1.0" "1.0" "1.0" ...
  ..$ idCounter     : int [1:11] 60714397 NA NA NA 60714399 NA NA NA NA 60714395 ...
  ..$ properties      :List of 11
$ numberVerification:'data.frame':    1 obs. of  1 variable:
  ..$ longNumber: num 2.81e+14
$ ndexStatus        :'data.frame':    1 obs. of  10 variables:
  ..$ externalId      : chr "eac8a4b8-6194-11e5-8ac5-06603eb7f303"
  ..$ creationTime    : num 1.44e+12
  ..$ modificationTime: num 1.44e+12
  ..$ visibility      : chr "PUBLIC"
  ..$ published       : logi FALSE
  ..$ nodeCount       : int 5
  ..$ edgeCount       : int 11
  ..$ owner           : chr "nci-pid"
  ..$ ndexServerURI   : chr "http://public.ndexbio.org"
  ..$ readOnly        : logi FALSE
$ @context          :'data.frame':    1 obs. of  23 variables:
  ..$ GENPEPT                     : chr "http://www.ncbi.nlm.nih.gov/protein/"
  ..$ NCBI GENE                   : chr "http://identifiers.org/ncbigene/"
  ..$ ENSEMBL                     : chr "http://identifiers.org/ensembl/"
  [...]
$ networkAttributes :'data.frame':    4 obs. of  2 variables:
  ..$ n: chr [1:4] "name" "description" "version" "ndex:sourceFormat"
  ..$ v: chr [1:4] "PLK3 signaling events" "This network ..." [...]
$ citations         :'data.frame':    4 obs. of  7 variables:
 ..$ @id           : int [1:4] 60714380 60714383 60714386 60714397
 ..$ dc:identifier : chr [1:4] "pmid:17264206" "pmid:14968113" "pmid:12242661" "pmid:11551930"
 ..$ dc:type       : chr [1:4] "URI" "URI" "URI" "URI"
 ..$ attributes    :List of 4 [...]
$ nodes             :'data.frame':    5 obs. of  2 variables:
 ..$ @id: int [1:5] 60714376 60714377 60714381 60714384 60714395
 ..$ n  : chr [1:5] "CCNE1" "PLK3" "MPIP3" "CHK2" ...
$ nodeAttributes    :'data.frame':    10 obs. of  4 variables:
 ..$ po: int [1:10] 60714376 60714376 60714377 60714377 60714381 60714381 60714384 60714384 60714395 60
  ..$ n : chr [1:10] "alias" "relatedTo" "alias" "relatedTo" ...
  ..$ v :List of 10
```

```
   .. ..$ : chr [1:6] "UniProt Knowledgebase:Q92501" "UniProt Knowledgebase:Q9UD21" ...
    .. ..$ : chr [1:98] "GENE ONTOLOGY:GO:0003713" "GENE ONTOLOGY:GO:0005515" ...
     [...]
  ..$ d : chr [1:10] "list_of_string" ...
$ edges             :'data.frame':    11 obs. of  4 variables:
  ..$ @id: int [1:11] 60714379 60714382 ...
  ..$ s  : int [1:11] 60714376 60714381 ...
  ..$ t  : int [1:11] 60714377 60714377 ...
  ..$ i  : chr [1:11] "neighbor-of" "neighbor-of" ...
$ edgeCitations     :'data.frame':    11 obs. of  2 variables:
  ..$ po       :List of 11
  .. ..$ : int 60714379
  .. ..$ : int 60714382
  [...]
..$ citations:List of 11
.. ..$ : int 60714380
.. ..$ : int 60714383
  [...]
$ status            :'data.frame':    1 obs. of  2 variables:
  ..$ error  : chr ""
  ..$ success: logi TRUE
- attr(*, "class")= chr [1:2] "RCX" "list"
```

The data.frames representing nodes and edges could look like this:

```
> rcx[["nodes"]]
   @id     n
 1 60714376 CCNE1
 2 60714377  PLK3
 3 60714381 MPIP3
 4 60714384  CHK2
 5 60714395   P53

> rcx[["edges"]]
    @id        s        t                          i
 1  60714379 60714376 60714377                neighbor-of
 2  60714382 60714381 60714377                neighbor-of
 3  60714385 60714384 60714377                neighbor-of
 4  60714388 60714377 60714376      controls-expression-of
 5  60714390 60714377 60714381 controls-phosphorylation-of
 6  60714392 60714377 60714381    controls-state-change-of
 7  60714393 60714377 60714384 controls-phosphorylation-of
 8  60714394 60714377 60714384    controls-state-change-of
 9  60714396 60714377 60714395 controls-phosphorylation-of
 10 60714398 60714377 60714395    controls-state-change-of
 11 60714399 60714377 60714395                neighbor-of
```

Value

returns object of class RCX if successfull, NULL otherwise

See Also

rcxgraph_fromRCX rcxgraph_toRCX rcx_toJSON

Examples

```
## Create an RCX object
rcx = rcx_new(c('@id'=1, n='Some Name', r='HGNC:Symbol'))
## Convert to JSON
json = rcx_toJSON(rcx)
## Convert it back from JSON
rcx = rcx_fromJSON(json)
```

---

rcx_new *Create a blank rcx object*

Description

This function generates a (blank) RCX object. For a valid RCX, at least one node has to be specified. Optional attributes are 'n' for names and 'r' for represents. Ids have to be unique (in nodes) and may not contain 'NA' values. For names and represents attributes, 'NA' values are allowed.

Usage

```
rcx_new(nodes = c(`@id` = 1))
```

Arguments

nodes	vector or data.frame (default: c('@id'=1)) node(s)

Value

RCX object

Examples

```
rcx = rcx_new()
rcx = rcx_new(c('@id'=1))                                #same as one before
rcx = rcx_new(nodes=c('@id'=1))                          #same as one before
rcx = rcx_new(data.frame('@id'=c(1), check.names=FALSE))     #same as one before
rcx = rcx_new(c('@id'=1, n='Some Name'))
rcx = rcx_new(c('@id'=1, n='Some Name', r='HGNC:Symbol'))
#same as one before:
rcx = rcx_new(data.frame('@id'=c(1),n=c('Some Name'), r=c('HGNC:Symbol'), check.names=FALSE))
rcx = rcx_new(data.frame('@id'=c(1,2,3),n=c('Some Name','And another name',NA),
                         r=c('HGNC:Symbol',NA,'UniProt:C3P0'), check.names=FALSE))
```

rcx_toJSON *Generate JSON data from RCX object*

Description

Generate JSON data from RCX object

Usage

```
rcx_toJSON(rcx, verbose = FALSE, pretty = FALSE)
```

Arguments

rcx	RCX object
verbose	logical; whether to print out extended feedback
pretty	logical; adds indentation whitespace to JSON output

Value

json jsonlite json object if successfull, NULL otherwise

See Also

rcxgraph_fromRCX rcxgraph_toRCX rcx_fromJSON

Examples

```
## Create an RCX object
rcx = rcx_new(c('@id'=1, n='Some Name', r='HGNC:Symbol'))
## Convert to JSON
json = rcx_toJSON(rcx)
```

rcx_toRCXgraph *Create RCXgraph object from RCX object*

Description

This function creates an RCXgraph object from a supplied RCX object. RCX objects store the CX data as a named list of data.frames containing metaData and all aspects of the network. The RCXgraph class inherits from igraph and contains the complete (R)CX information as graph, node and edge attributes. All igraph functionality is available, e.g. access nodes and edges of igraph g via V(g) and E(g) and their attributes via V(g)$attribute

Usage

```
rcx_toRCXgraph(
  rcx,
  idAsVertexName = FALSE,
  idAsEdgeName = FALSE,
  verbose = FALSE
)

rcxgraph_fromRCX(
  rcx,
  idAsVertexName = FALSE,
  idAsEdgeName = FALSE,
  verbose = FALSE
)
```

Arguments

rcx	RCX object
idAsVertexName	logical; whether the ndex node id ("@id") should be used as name for the igraph node (i.e. vertex). By default the "n"ame property is used
idAsEdgeName	logical; whether the ndex edge id ("@id") should be used as name for the igraph edge. By default the "i"nteraction property is used
verbose	logical; whether to print out extended feedback

Details

The following rules apply to convert from RCX to RCXgraph:

- nodes receive their name from RCX$node$n. If idAsVertexName is TRUE, the "@id" value is used as name. All other information in aspects node and nodeAttributes are saved as vertex_attr, access via V(g). Data goes from long format (column n containing attribute name and column v containing attribute value) to wide format (columns for each unique n with cells contianing v).
- edges are connected via their "s"ource and "t"arget fields. The "@id" and "i"nteraction attribute are stored as is and all edgeAttributes are saved as edge_attr, access via E(g). Data goes from long format (column n containing attribute name and column v containing attribute value) to wide format (columns for each unique n with cells contianing v).
- all other aspect data is stored as graph attributes, access via g$<aspectName>

The following rules apply to convert RCXgraph back to RCX:

- Two vertex attributes "n" and "@id" have to be present in RCXgraph! Those two are mandatory RCX node properties!
- The igraph vertex name is ignored for the conversion! If the name is needed, adjust manually, e.g.: V(RCXgraph)$n <- V(RCXgraph)$name
- The edge attribute "@id" has to be present in RCXgraph! This is a mandatory RCX edge property!

An RCXgraph object could look like this:

```
## Get some network...
> ndexcon = ndex_connect()
> rcx = ndex_get_network(ndexcon,"dd268e2f-fd4d-11e7-adc1-0ac135e8bacf")

> summary(rcxgraph)
IGRAPH f99ed1e DN-- 30 218 --
+ attr: metaData (g/x), numberVerification (g/x), ndexStatus (g/x), provenanceHistory (g/x), networkAtt
| (g/x), cyHiddenAttributes (g/x), status (g/x), name (v/c), @id (v/n), n (v/c), Basal (v/c), avg_PPR (v
| (v/c), Her2 (v/c), name (e/c), @id (e/n), i (e/c), NAME (e/c), strength (e/c), interaction (e/c)

## Attributes marked with "(g/x)" are graph attributes, "(v/x)" and "(e/x)" correspond to vertex and edg
## Accessing a graph attribute, e.g. "metaData"
rcxgraph$metaData
  consistencyGroup elementCount  lastUpdate          name properties version idCounter
1              1          1 1.516391e+12       ndexStatus      NULL    1.0       NA
2              1          1 1.516391e+12  provenanceHistory      NULL    1.0       NA
3              1         30          NA            nodes      NULL    1.0      826
4              1        218          NA            edges      NULL    1.0      827
5              1         11          NA  networkAttributes      NULL    1.0       NA
6              1        210          NA    nodeAttributes      NULL    1.0       NA
7              1        654          NA    edgeAttributes      NULL    1.0       NA
8              1         30          NA   cartesianLayout      NULL    1.0       NA
9              1          3          NA cyVisualProperties      NULL    1.0       NA
10             1          1          NA cyHiddenAttributes      NULL    1.0       NA


## The vertices:
> V(rcxgraph)[[]]
+ 30/30 vertices, named, from f99ed1e:
   name X.id    n        Basal       avg_PPR         LumA         LumB      ANOVA_FDR        ANOVA_p
1   ABL1  332  ABL1 2.10282304283E-4 0.00153239525422 2.40855043132E-4 0.00243281556888  0.0126090186
2   JAK2  331  JAK2 0.00818834880039 0.00229451204468 0.00112294137386 5.02052428871E-4  2.1649538801
3 NOTCH4  330 NOTCH4 0.00271376234049 0.00146318626692 9.53762145731E-4 0.00122487574678  2.164953880
4  KMT2C  329  KMT2C 0.00628341277798 0.00228045774193 0.00156001096343 0.00150488341005  0.022707788
5 NOTCH1  328 NOTCH1 0.00839759047274  0.0185377425722  0.0238783637494  0.0174878270867 5.65380895701
6 ...

## Display the igraph vertex names
> V(rcxgraph)$name
 [1] "ABL1"  "JAK2"  "NOTCH4" "KMT2C"  "NOTCH1" "INPPL1" "SPOP"  "AKT1"  "MYC"   "GATA3"  "MET"   "PTPR
[16] "IBSP"  "ERBB2"  "PIK3CA" "EGFR"  "TLN1"  "MYCN"  "KRAS"  "CDH1"  "TP53"  "CCNE1"  "CCND1"  "CARD

## The igraph vertex names equal to the ndex node names by default
> V(rcxgraph)$n
 [1] "ABL1"  "JAK2"  "NOTCH4" "KMT2C"  "NOTCH1" "INPPL1" "SPOP"  "AKT1"  "MYC"   "GATA3"  "MET"   "PTPR
[16] "IBSP"  "ERBB2"  "PIK3CA" "EGFR"  "TLN1"  "MYCN"  "KRAS"  "CDH1"  "TP53"  "CCNE1"  "CCND1"  "CARD
```

```
## If in \link{rcx_toRCXgraph} or \link{rcxgraph_fromRCX} \quote{idAsVertexName} is set, the igraph ver
## Note: "@id" is displayed in the vertex summary as "X.id", but can be accessed using "@id"
> V(rcxgraph)$'@id'
[1] 332 331 330 329 328 327 326 325 324 323 322 321 320 319 318 317 316 315 314 313 312 311 310 309 308 307


## The edges:
> E(rcxgraph)[[]]
+ 218/218 edges from f99ed1e (vertex names):
    tail  head tid hid       name X.id         i                  NAME       strength   interaction
1   ABL1  EGFR  1  19 interacts with  550 interacts with   ABL1 (interacts with) EGFR  2.6501871987E-8
2   ABL1 GATA3  1  10 interacts with  549 interacts with   ABL1 (interacts with) GATA3 0.0017942365068
3   ABL1 INPPL1  1  6 interacts with  548 interacts with  ABL1 (interacts with) INPPL1 0.0053112520138
4   ABL1  JAK2  1  2 interacts with  547 interacts with   ABL1 (interacts with) JAK2 0.00668830637406
5   ABL1   MET  1  11 interacts with  546 interacts with   ABL1 (interacts with) MET 2.37014815511E-5
6 ...

## Display the igraph edge names
> E(rcxgraph)$name
[1] "interacts with" "interacts with" "interacts with" "interacts with" "interacts with" "interacts wit
[8] ...

## The igraph edge names equal to the ndex edge "i"nteraction by default
> E(rcxgraph)$i
[1] "interacts with" "interacts with" "interacts with" "interacts with" "interacts with" "interacts wit
[8] ...

## If in \link{rcx_toRCXgraph} or \link{rcxgraph_fromRCX} \quote{idAsEdgeName} is set, the igraph edge
## Note: "@id" is displayed in the edge summary as "X.id", but can be accessed using "@id"
> E(rcxgraph)$'@id'
 [1] 550 549 548 547 546 545 544 543 542 541 540 539 538 537 536 535 534 533 532 531 530 529 528 527 526 525
[30] ...
```

Value

returns object of class RCXgraph if successfull, NULL otherwise

See Also

rcxgraph_toRCX rcx_fromRCXgraph rcx_fromJSON rcx_toJSON RCX igraph

Examples

```
## Create an RCX object
rcx = rcx_new(c('@id'=1, n='Some Name', r='HGNC:Symbol'))

## Convert to RCXgraph
rcxgraph = rcxgraph_fromRCX(rcx)
## or
rcxgraph = rcx_toRCXgraph(rcx)
```

```
## Convert RCXgraph back to RCX
rcx = rcx_fromRCXgraph(rcxgraph)
## or
rcx = rcxgraph_toRCX(rcxgraph)
```

`rcx_updateMetaData` *Updating the meta-data of an RCX object*

Description

Updating the meta-data of an RCX object

Usage

```
rcx_updateMetaData(
  rcx,
  mandatoryAspects = c("nodes"),
  excludeAspects = c("metaData", "numberVerification", "status"),
  force = FALSE,
  verbose = FALSE
)
```

Arguments

`rcx`	RCX object
`mandatoryAspects`	character vector; Aspects, that are mandatory for a valid RCX object (by default: "nodes")
`excludeAspects`	character vector; Aspects, that are excluded for generating metaData (by default: "metaData", "numberVerification" and "status")
`force`	logical; force the creation of new metaData (even if the RCX object already contains metaData)
`verbose`	logical; whether to print out extended feedback

Details

For a given RCX object the meta-data is updated, i.e. the counted elements and id counter are updated. If an aspect was added/removed, it will also added/removed from the meta-data. If mandatory aspects (specified in mandatoryAspects parameter) are missing in the RCX object, an error is thrown.

Value

RCX object

Examples

```
## Create an RCX object
rcx = rcx_new(c('@id'=1, n='Some Name', r='HGNC:Symbol'))
## update meta-data
rcx = rcx_updateMetaData(rcx)
# or with explicitly set default values
rcx = rcx_updateMetaData(rcx, mandatoryAspects=c('nodes'),
                         excludeAspects=c("metaData", "numberVerification", "status"),
                         force=FALSE, verbose=FALSE)
```

Index