ndex2 Documentation

Release 2.0.1

Dexter Pratt, Aaron Gary & Jing Chen

Apr 23, 2019

Contents

Contents:

CHAPTER 1

NiceCXNetwork module

The NiceCXNetwork class provides a data model for working with NDEx networks. Methods are provided to add nodes, edges, node attributes, edge attributes, etc. Once a NiceCXNetwork data object is populated it can be saved to the NDEx server by calling either **upload_to()** to create a new network or **update_to()** to update an existing network.

To see deprecated methods go to *Deprecated NiceCXNetwork methods*

1.1 Methods for building niceCX

see also this notebook

1.1.1 Node methods

1.1.2 Edge methods

1.1.3 Network methods

1.2 Methods for accessing niceCX properties

see also this notebook

1.2.1 Node methods

1.2.2 Edge methods

1.2.3 Network methods

1.3 Misc niceCX methods

1.4 Deprecated NiceCXNetwork methods

1.5 Supported data types

The following data types are supported in methods that accept **type**

Example:

```
set_edge_attribute(0, 'weight', 0.5, type='double')
```

- string
- double
- boolean
- integer
- long
- list_of_string
- list_of_double
- list_of_boolean
- list_of_integer
- list_of_long

1.6 Methods for creating niceCX from other data models

ndex2.**create_nice_cx_from_raw_cx**(*cx*)
Create a NiceCXNetwork from a CX json object. (see http://www.home.ndexbio.org/data-model)

Parameters **cx** – a valid CX document

Returns NiceCXNetwork

ndex2.**create_nice_cx_from_file**(*path*)
Create a NiceCXNetwork from a file that is in the CX format.

Parameters **path** – the path of the CX file

Returns NiceCXNetwork

ndex2.**create_nice_cx_from_networkx**(*G*)
Creates a NiceCXNetwork based on a networkx graph. The resulting NiceCXNetwork contains the nodes edges

and their attributes from the networkx graph and also preserves the graph 'pos' attribute as a CX cartesian coordinates aspect. Node name is taken from the networkx node id. Node 'represents' is taken from the networkx node attribute 'represents'

Parameters **G** (*networkx graph*) – networkx graph

Returns NiceCXNetwork

Return type NiceCXNetwork

ndex2.**create_nice_cx_from_pandas**(*df*, *source_field=None*, *target_field=None*, *source_node_attr=[]*, *target_node_attr=[]*, *edge_attr=[]*, *edge_interaction=None*, *source_represents=None*, *target_represents=None*)

Create a NiceCXNetwork from a pandas dataframe in which each row specifies one edge in the network.

If only the df argument is provided the dataframe is treated as 'SIF' format, where the first two columns specify the source and target node ids of the edge and all other columns are ignored. The edge interaction is defaulted to "interacts-with"

If both the source_field and target_field arguments are provided, the those and any other arguments refer to headers in the dataframe, controlling the mapping of columns to the attributes of nodes, and edges in the resulting NiceCXNetwork. If a header is not mapped the corresponding column is ignored. If the edge_interaction is not specified it defaults to "interacts-with"

Parameters

- **df** – pandas dataframe to process
- **source_field** – header name specifying the name of the source node.
- **target_field** – header name specifying the name of the target node.
- **source_node_attr** – list of header names specifying attributes of the source node.
- **target_node_attr** – list of header names specifying attributes of the target node.
- **edge_attr** – list of header names specifying attributes of the edge.
- **edge_interaction** – the relationship between the source node and the target node, defaulting to "interacts-with"

Returns NiceCXNetwork

ndex2.**create_nice_cx_from_server**(*server*, *username=None*, *password=None*, *uuid=None*)

Create a NiceCXNetwork based on a network retrieved from NDEx, specified by its UUID. If the network is not public, then username and password arguments for an account on the server with permission to access the network must be supplied.

Parameters

- **server** – the URL of the NDEx server hosting the network.
- **username** – the user name of an account with permission to access the network.
- **password** – the password of an account with permission to access the network.
- **uuid** – the UUID of the network.

Returns NiceCXNetwork

1.7 Client access to NDEx server API

class ndex2.client.**Ndex2**(*host=None*, *username=None*, *password=None*, *update_status=False*, *debug=False*, *user_agent=''*)

A class to facilitate communication with an NDEx server.

If host is not provided it will default to the NDEx public server. UUID is required

add_networks_to_networkset(*set_id*, *networks*)

Add networks to a network set. User must have visibility of all networks being added

Parameters

- **set_id** (*basestring*) – network set id
- **networks** (*list of strings*) – networks that will be added to the set

Returns None

Return type None

create_networkset(*name*, *description*)

Creates a new network set

Parameters

- **name** (*string*) – Network set name
- **description** (*string*) – Network set description

Returns URI of the newly created network set

Return type string

delete_network(*network_id*, *retry=5*)

Deletes the specified network from the server

Parameters

- **network_id** (*string*) – Network id
- **retry** (*int*) – Number of times to retry if deleting fails

Returns Error json if there is an error. Blank

Return type string

delete_networks_from_networkset(*set_id*, *networks*, *retry=5*)

Removes network(s) from a network set.

Parameters

- **set_id** (*basestring*) – network set id
- **networks** (*list of strings*) – networks that will be removed from the set
- **retry** (*int*) – Number of times to retry

Returns None

Return type None

get_neighborhood(*network_id*, *search_string*, *search_depth=1*, *edge_limit=2500*)

Get the CX for a subnetwork of the network specified by UUID network_id and a traversal of search_depth steps around the nodes found by search_string.

Parameters

- **network_id** (*str*) – The UUID of the network.
- **search_string** (*str*) – The search string used to identify the network neighborhood.
- **search_depth** (*int*) – The depth of the neighborhood from the core nodes identified.
- **edge_limit** (*int*) – The maximum size of the neighborhood.

Returns The CX json object.

Return type response object

get_neighborhood_as_cx_stream(*network_id*, *search_string*, *search_depth=1*, *edge_limit=2500*, *error_when_limit=True*)

Get a CX stream for a subnetwork of the network specified by UUID network_id and a traversal of search_depth steps around the nodes found by search_string.

Parameters

- **network_id** (*str*) – The UUID of the network.
- **search_string** (*str*) – The search string used to identify the network neighborhood.
- **search_depth** (*int*) – The depth of the neighborhood from the core nodes identified.
- **edge_limit** (*int*) – The maximum size of the neighborhood.
- **error_when_limit** (*boolean*) – Default value is true. If this value is true the server will stop streaming the network when it hits the edgeLimit, add success: false and error: "EdgeLimitExceeded" in the status aspect and close the CX stream. If this value is set to false the server will return a subnetwork with edge count up to edgeLimit. The status aspect will be a success, and a network attribute {"EdgeLimitExceeded": "true"} will be added to the returned network only if the server hits the edgeLimit..

Returns The response.

Return type

response object

get_network_as_cx_stream(*network_id*)

Get the existing network with UUID network_id from the NDEx connection as a CX stream.

Parameters **network_id** (*str*) – The UUID of the network.

Returns The response.

Return type

response object

get_network_ids_for_user(*username*)

Get the network uuids owned by the user

Parameters **username** (*str*) – users NDEx username

Returns list of uuids

get_network_set(*set_id*)

Gets the network set information including the list of networks

Parameters **set_id** (*basestring*) – network set id

Returns network set information

Return type dict

get_network_summary(*network_id*)
Gets information about a network.

Parameters **network_id** (*str*) – The UUID of the network.

Returns Summary

Return type dict

get_sample_network(*network_id*)
Gets the sample network

Parameters **network_id** (*string*) – Network id

Returns Sample network

Return type list of dicts in cx format

get_task_by_id(*task_id*)
Retrieves a task by id

Parameters **task_id** (*string*) – Task id

Returns Task

Return type dict

get_user_by_username(*username*)
Gets the user id by user name

Parameters **username** (*string*) – User name

Returns User id

Return type string

get_user_network_summaries(*username*, *offset=0*, *limit=1000*)
Get a list of network summaries for networks owned by specified user. It returns not only the networks that the user owns but also the networks that are shared with them directly.

Parameters

- **username** (*str*) – the username of the network owner
- **offset** (*int*) – the starting position of the network search
- **limit** –

Returns list of uuids

Return type list

grant_network_to_user_by_username(*username*, *network_id*, *permission*)
Grants permission to network for the given user name

Parameters

- **username** (*string*) – User name
- **network_id** (*string*) – Network id
- **permission** (*string*) – Network permission

Returns Result

Return type dict

grant_networks_to_group(*groupid*, *networkids*, *permission='READ'*)
Set group permission for a set of networks

Parameters

- **groupid** (*string*) – Group id
- **networkids** (*list*) – List of network ids
- **permission** (*string*) – Network permission

Returns Result

Return type dict

grant_networks_to_user(*userid*, *networkids*, *permission='READ'*)
Gives read permission to specified networks for the provided user

Parameters

- **userid** (*string*) – User id
- **networkids** (*list of strings*) – Network ids
- **permission** (*string (default is READ)*) – Network permissions

Returns none

Return type none

make_network_private(*network_id*)
Makes the network specified by the network_id private.

Parameters **network_id** (*str*) – The UUID of the network.

Returns The response.

Return type

response object

make_network_public(*network_id*)
Makes the network specified by the network_id public.

Parameters **network_id** (*str*) – The UUID of the network.

Returns The response.

Return type

response object

save_cx_stream_as_new_network(*cx_stream*, *visibility=None*)
Create a new network from a CX stream.

Parameters

- **cx_stream** (*BytesIO*) – IO stream of cx
- **visibility** (*string*) – Sets the visibility (PUBLIC or PRIVATE)

Returns Response data

Return type string or dict

save_new_network(*cx*, *visibility=None*)
Create a new network (cx) on the server

Parameters

- **cx** (*list of dicts*) – Network cx
- **visibility** (*string*) – Sets the visibility (PUBLIC or PRIVATE)

Returns Response data

Return type string or dict

search_networks(*search_string=''*, *account_name=None*, *start=0*, *size=100*, *include_groups=False*)
Search for networks based on the search_text, optionally limited to networks owned by the specified account_name.

Parameters

- **search_string** (*str*) – The text to search for.
- **account_name** (*str*) – The account to search
- **start** (*int*) – The number of blocks to skip. Usually zero, but may be used to page results.
- **size** (*int*) – The size of the block.
- **include_groups** –

Returns The response.

Return type

response object

set_network_properties(*network_id*, *network_properties*)
Sets network properties

Parameters

- **network_id** (*string*) – Network id
- **network_properties** (*list*) – List of NDEx property value pairs

Returns

Return type

set_network_system_properties(*network_id*, *network_properties*)
Set network system properties

Parameters

- **network_id** (*string*) – Network id
- **network_properties** (*dict of NDEx network property value pairs*) – Network properties

Returns Result

Return type dict

set_read_only(*network_id*, *value*)
Sets the read only flag on the specified network

Parameters

- **network_id** (*string*) – Network id
- **value** (*bool*) – Read only value

Returns Result

Return type dict

update_cx_network(*cx_stream*, *network_id*)
Update the network specified by UUID network_id using the CX stream cx_stream.

Parameters

- **cx_stream** – The network stream.
- **network_id** (*str*) – The UUID of the network.

Returns The response.

Return type

response object

update_network_group_permission(*groupid*, *networkid*, *permission*)
Updated group permissions

Parameters

- **groupid** (*string*) – Group id
- **networkid** (*string*) – Network id
- **permission** (*string*) – Network permission

Returns Result

Return type dict

update_network_profile(*network_id*, *network_profile*)
Updates the network profile 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. The network profile attributes that can be updated by this method are: 'name', 'description' and 'version'.

Parameters

- **network_id** (*string*) – Network id
- **network_profile** (*dict*) – Network profile

Returns

Return type

update_network_user_permission(*userid*, *networkid*, *permission*)
Updated network user permission

Parameters

- **userid** (*string*) – User id
- **networkid** (*string*) – Network id
- **permission** (*string*) – Network permission

Returns Result

Return type dict

CHAPTER 2

ndex2

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

n

Index