NDEx 2.5 REST API

Last updated: April 14th, 2023

Overview

The NDEx 2.5 API is the currently supported REST API and is available at https://www.ndexbio.org.

The older v1.3 API is deprecated and we strongly encourage you to migrate your applications to use the latest version.

The non-secure "http" protocol is also deprecated and we will stop supporting it soon.

The CX Network Exchange Format

Networks sent or received using this API are in CX format. Conversion between CX and other formats is intended to be handled by separate services or libraries. The CX format is described in the CX Data Model (../data-model/) document available on this website.

Best Practices for Using the API in Applications

For most purposes, developers are advised to use client libraries that provide convenient access to commonly methods, especially those for network I/O, search, and query.

For advanced applications that attempt to manage users, groups, permissions, tasks, and requests, the NDEx team would like to work with you, initially using our TEST SERVER (http://test.ndexbio.org).

Available Libraries:

- Python Client, also available via PyPi.
- Java Client and Java Object Model
- R Client, also available via Bioconductor.

For links to code repositories, tutorials and other technical documentation, please refer to the DEVELOPER'S README!!! (../readme-developers-best-practices/) page.

API Design Guidelines Used in NDEx 2.5

1 – All the synchronized (blocking) functions use these conventions:

a. Creation: API functions that create resource objects in NDEx server

i. HTTP Method: POST

ii. HTTP return code: 201 on success.

iii. Location field will be populated with the relative url for retrieving the newly created object.

iv. The body of the return is the full URI of the created object.

b. Update: API functions that modify existing resource objects in NDEx server

i. HTTP Method: PUT

ii. HTTP return code: 204 on success

c. Deletion: API functions that delete resource objects in NDEx server

i. HTTP Method: DELETE

ii. HTTP return code: 204 on success

d. Query/retrieval:

i. In general, functions that query or retrieve information from the NDEx server use GET.

ii. For Batch retrieval of resource objects and search functions, we use POST method.

2 – Asynchronized (non-blocking) functions use this convention:

a. Return code: 202 on success

b. “Location” field in the response header will be populated with the relative URL of the task object.

3 – Functions that can potentially return large amount of data provide optional parameters “start” and “size” in their URLs to support paging in client applications.

a. start=n specifies that the result is the *nth* page of the requested data. The default value is 0.

b. size=n specifies the number of data items in each page. The default value is 100.

4 – Authentication:

a. The public NDEx server supports BASIC AUTH as the authentication method.

b. In this API document, “Authentication: Not required” means the function returns the same result regardless of whether BASIC AUTH credentials are provided in the request or whether the credential are successfully authenticated.

c. “Authentication: Required” means that the request must provide BASIC AUTH credentials that are successfully authenticated. Otherwise a 401 UNAUTHORIZED code will be returned.

d. “Authentication: Optional” means the API function supports both authenticated and anonymous requests, but the result of the function may be different for authenticated requests vs. anonymous requests.

e. Some of the GET functions in the Network Category supports an optional parameter called accessKey. This accessKey allows users to bypass the authentication and have the read access to that resource directly. The network accessKey can be turned on and off using the function "Disable/enable Access Key on Network".

f. The NDEx server also supports Google OAuth as another authentication method. To use Google OAuth:

i. Get an ID_token from Google in your program and add the ID_token string to the Authorization header in your HTTP calls.

ii. The format of your Authorization value is ‘Bearer id_token_string’

iii. Please Contact Us (../contact-us) to register your application and obtain the required client ID.

5 – Errors:

a. Unless otherwise stated, NDEx API functions return an HTTP return code 4xx or 500 on errors.

b. The body of these error responses are a json value that includes the error message from the server.

Public API Functions Categorized by Resource Type

Admin

Get Server Status

/admin/status?format={full|standard}

Method: GET

Authentication: Not required

Get the current status of the server. Use this function to check if the server is running and which version it is. The default value for parameter format is ‘standard’.

An example of returned object:

```
{ "networkCount": 1321,
  "userCount": 12,
  "groupCount":0,
  "message":"Online",
  "properties": { "ServerVersion":"2.1",
  "ServerResultLimit":"10000"
 }
}
```


User

Get User's Membership in Group

/user/{userid}/membership?groupid={groupid}

Method: GET

Authentication: Required

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.

Get User's Group Memberships

/user/{userid}/membership?type={membershiptype}&start={startPage}&size={pageSize}

Method: GET

Authentication: Optional

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.

Get User's Permission for Network

/user/{userid}/permission?networkid={networkId}&directonly={true|false}

Method: GET

Authentication: Required

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.
- If directonly is set to true, permissions granted through groups are not included in the result. the default value for directonly is false.

Get User's Account Page Networks

/user/{userid}/networksummary?offset={offset}&limit={limit}

Method: GET

Authentication: Required

Description:

- This is a function designed to support "My Account" pages in NDEx applications. It returns a list of NetworkSummary objects to display.
- It returns not only the networks that the user owns but also the networks that are shared with them directly.
- Userid must be the authenticated user's UUID.

Get All Network Sets owned by a user

/user/{userid}/networksets?offset={offset}&limit={limit}&summary={true|false}&showcase={true|false}

Method: GET

Authentication: Not required

Description: Get a list of network sets that are owned by a user.

Parameters:

- summary: When this parameter is set to true, the server will not return the list of network IDs in this network set. It can significantly reduce the response time from the server, if some of the networks sets have large amount of networks in them. Default value is false.
- showcase: When this parameter is set to true, only showcased network sets will be returned. Default value is false.

Group

Create Group

/group

Method: POST

Authentication: required

Description:

- Create a group owned by the authenticated user based on the supplied group JSON object.
- Errors if the group name specified in the JSON object is not valid or is already in use.
- Structure of a group object is:

```
{
"properties": {},
  "groupName": "Melanoma Group",
  "image": “https://example.com/image/group1.jpg”,
  "website": "ucsd.edu",
  "description": "description of group"
}
```

Update Group

/group/{groupid}

Method: PUT

Authentication: required

Description:

- Updates the group metadata corresponding to the POSTed group JSON object.
- Errors if the JSON object does not specify the group id or if no group is found by that id.

Delete Group

/group/{groupid}

Method: DELETE

Authentication: required

Description:

- Delete the group specified by groupId.
- Errors if the group is not found or if the authenticated user does not have authorization to delete the group.

Get a Group

/group/{groupid}

Method: GET

Authentication: Not required

Description:

- Returns a group JSON structure for the group specified by groupId. The returned group object has these attributes:

Attribute	Data type	Description
externalId	string	uuid of this group.
groupName	string	A short name of this group
creationTime	Timestamp	Time when this group was created in NDEx
modificationTime	Timestamp	Time when this group was last modified.
properties	object	Properties of of this group.
image	string	URL of this group's image.
website	string	URL of this group's web site
description	string	Short description of this group.

- Errors if the group is not found.

Add or Update a Group Member

/group/{groupid}/membership?userid={userid}&type={GROUPADMIN|MEMBER}

Method: PUT

Authentication: required

Description:

- Updates the membership corresponding to the GroupMembership type specified in the URL parameter.
- Errors if GroupMembership type, groupUUID or userid is not provided.
- Errors if the authenticated user does not have admin permissions for the group.
- Errors if the change would leave the group without an Admin member.

Remove a Group Member

/group/{groupid}/membership?userid={userid}

Method: DELETE

Authentication: Required

Description:

- Removes the member specified by userUUID from the group specified by groupUUID.
- Errors if the group or the user is not found.
- Errors if the authenticated user is not authorized to edit the group.
- Errors if removing the member would leave the group with no Admin member.

Get Members of a Group

/group/{groupid}/membership?type={membershiptype}&start={startPage}&size={pageSize}

Method: GET

Authentication: Optional

Description:

- This function returns user membership JSON objects of type *membershiptype* for a group. If the 'type' parameter is omitted, all membership types will be returned.
- The structure of membership object is:

Attribute	Data type	Description
permissions	string	Type of membership a user has. Should be one of these values: "GROUPADMIN" or "MEMBER".
memberUUID	string	Uuid of this member
resourceUUID	string	Uuid of this group
modificationTime	Timestamp	Time when this group was last modified.
memberAccountName	string	userName of the member
resourceName	string	Name of the group

Get Network Permissions of a Group

/group/{groupid}/permission?permission={permission}&start={startPage}&size={pageSize}

Method: GET

Authentication: Optional

Description:

- Returns network permissions for the specified group.

- Returned object is a map in which keys are network UUIDs and values are the permission to the network assigned to the specified group.

Get Group Permission for a Specific Network

/group/{groupid}/permission?networkid={networkid}

Method: GET

Authentication: Required

Description :

- Returns the explicit permission the specified group has on the specified network, which is either READ or WRITE.
- Note that this function always requires authentication in order to ensure that only *members* of a group can obtain group-network permissions.

Task

Get User's Tasks

/task?status={status}&start={startPage}&size={pageSize}

Method: GET

Authentication: Required

Description:

- Returns a JSON array of Task objects owned by the authenticated user with the specified status.
- The array is ordered by task creation time.
- If no status is specified, all tasks owned by this user will be returned. Valid task status are: QUEUED, PROCESSING, COMPLETED, FAILED.
- A task object has this structure:

Attribute	Data type	Description
externalId	string	uuid of this task.
creationTime	Timestamp	Time when this task was created in NDEx
modificationTime	Timestamp	Time when this task was last modified.
attributes	object	Additional attributes of of this task.
format	string	Format of the exported file if this is a network export task.

description	string	Short description of this task
taskType	string	Type of this task
status	string	Status of this task
taskOwnerId	string	UUID of this task's owner
message	string	Error message if this task failed.
resource	uuid	UUID of the resource which this task operates on.
startTime	Timestamp	Time when this task started running
finishTime	Timestamp	Time when this task stopped running

Get a Task by Task UUID

/task/{taskid}

Method: GET
Authentication: Required
Description:

- Returns a JSON task object for the task specified by taskId.
- Errors if no task found or if the authenticated user does not own the specified task.

Download exported file by Task UUID

/task/{taskid}/file?download=true

Method: GET
Authentication: Required
Description:

- Returns the file exported by the given task.
- Errors if no task found or if the authenticated user does not own the specified task.

Delete a Task

/task/{taskid}

Method: DELETE
Authentication: Required
Description:

- Deletes the task specified by taskId.
- Errors if no task found or if authenticated user does not own the specified task.

Network

Create a CX Network

/network?visibility=PRIVATE|PUBLIC

Method: POST

Authentication: Required

Description:

- Create a network from CX data.
- If the POSTed content-type is multipart/form-data, the input CX data is expected to be in the CXNetworkStream field.
- If the POSTed content-type is application/json, user can post the CX data directly to the payload.
- Users can use the optional visibility parameter to set the visibility of a network when uploading it. The default value of this parameter is PRIVATE.

Update a Network

/network/{networkid}

Method: PUT

Authentication: required

Description:

- Update the specified network with new content, based on CX data.
- If the content type is multipart/form-data, the CX data is in the CXNetworkStream field of PUTed multipart/form-data. If the content type is application/json, the CX data is in the payload.
- Errors if the CX data is not provided
- Errors if the UUID does not correspond to an existing network on the NDEx Server for which the authenticated user owns or for which they have WRITE permission.
- Errors if the CX data is larger than the maximum size of a network the current NDEx server allows.

Delete a Network

/network/{networkid}

Method: DELETE

Authentication: Required

Description:

- Deletes the network specified by networkId.

- There is no method to undo a deletion, so care should be exercised.
- The specified network must be owned by the authenticated user.

Get a Network Summary

/network/{networkid}/summary?accesskey={accessKey}

Method: Get

Authentication: Optional

Description:

- Retrieves a NetworkSummary JSON object based on the network specified by networkId.
- A NetworkSummary object is a subset of a network object. It is used to convey basic information about a network in this API.

Attribute	Datatype	Description
name	string	Name or title of the network, not unique, same meaning as dc:title
description	string	Text description of the network, same meaning as dc:description
creationTme	timeStamp	Time at which the network was created
modificationTime	timeStamp	Time at which the network was last modified
visibility	string	One of PUBLIC, PRIVATE. PUBLIC means it can be found or read by anyone, including anonymous users. PRIVATE is the default, means that it can only be found or read by users according to their permissions.
version	string	Format is not controlled but best practice is to use string conforming to Semantic Versioning.
nodeCount	integer	the number of node objects in the network
edgeCount	integer	the number of edge objects in the network
properties	list	List of NDExPropertyValuePair objects: describes properties of the network.
externalId	string	Uuid of this network
ownerUUID	string	Uuid of this networks owner

isReadOnly	boolean	True if this network is marked as readonly in NDEx.
subnetworkIds	list	List of integers which are identifiers of subnetworks.
errorMessage	string	If this network is not a valid CX network, this field holds the error message produced by the CX network validator.
isValid	boolean	True if this network is a valid CX network.
owner	string	userName of the network owner.
indexed	boolean	True if the network needs to be indexed.
completed	boolean	True means all pending operation on this network has been finished.
warnings	list of string	If there are potential errors in the network, this field holds the warning message produced by the CX network validator.
isShowcase	boolean	True if this network is showcased in the owner's account.
isCertified	boolean	True if this is a published network in NDEx, with a DOI assigned and a valid publication reference.
hasLayout	boolean	True if the network has coordinates on its nodes.
hasSample	boolean	True if the network has a sample network.

- Errors if the network is not found or if the authenticated user does not have READ permission for the network.
- Anonymous users can only access networks with visibility = PUBLIC.

Get Complete Network in CX format

/network/{networkid}?accesskey={accessKey}

Method: GET

Authentication: Optional

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.

Set Network System Properties

/network/{networkId}/systemproperty

Method: PUT

Authentication: Required

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.
- Sets the system property specified in the PUT data for the network specified by networkid.
- As of NDEx V2.0 the supported system properties are:
  - readOnly: boolean.
  - visibility: supported values are PUBLIC or PRIVATE.
  - showcase: boolean. Authenticated users 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.
- PUT data format: {property: value} such as { "readOnly": true} or {"visibility": "PUBLIC"} or a JSON object with both properties

Get Network Sample

/network/{networkid}/sample?accesskey={accessKey}

Method: GET

Authentication: Optional

Description:

- Returns a sample subnetwork of the network specified by networkid.
- The default sample network is created based on an arbitrary selection of 500 edges, created by the NDEx server for networks larger than 500 edges.
- Alternatively, if a sample network has been explicitly stored using the setNetworkSample method, that sample network will be returned.
- Returns ObjectNotFound exception when no sample file was found:
  - Network specified by networkid has 500 edges or fewer.
  - The sample network is not yet generated by the server at the time of the request.
  - The sample network has been set to null by the network owner.
- Errors if the authenticated user making the request does not have WRITE or ADMIN permissions to the specified network.
- Errors if networkid does not correspond to an existing network.

Set Sample Network

/network/{networkid}/sample

Method: PUT

Authentication: Required

Description:

- Sets the sample network for the network specified by *networkid*. The sample network is specified by CX data in the PUT data of the request.
- Errors if the authenticated user does not have ADMIN permissions to the specified network.
- Errors if networkid does not correspond to an existing network.

Update Network Profile

/network/{networkid}/profile

Method: PUT

Authentication: Required

Description:

- Updates the profile information of the network specified by *networkid* based on a POSTed JSON object specifying the attributes to update.
- 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'.

Set Network Properties

/network/{networkid}/properties

Method: PUT

Authentication: Required

Description:

- Updates the NetworkAttributes aspect the network specified by 'networkId' based on the list of NdexPropertyValuePair objects in the PUT data.
- NdexPropertyValuePair object has these attributes:

Attribute	Datatype	Description
subNetworkId	string	Identifier of the subnetwork to which the property applies.
predicateString	string	Name of this property

dataType	string	Data type of this property. Its value has to be one of the attribute data types that CX supports.
value	string	A string representation of the property value.

- Errors if the authenticated user does not have ADMIN permissions to the specified network.
- Errors if networkid does not correspond to an existing network.

Get Network Provenance

/network/{networkid}/provenance?accesskey={accessKey}

Method: GET

Authentication: Optional

Description:

- Returns the Provenance aspect of the network specified by *networkid*.
- The returned value is a JSON ProvenanceEntity object which in turn contains a tree-structure of ProvenanceEvent and ProvenanceEntity objects that describe the provenance history of the network.
- See the document NDEx Provenance History for a detailed description of this structure and best practices for its use.
- Errors if *networkid* does not correspond to an existing network.
- Provenance History Structure
    - A provenance history is a tree structure containing ProvenanceEntity and ProvenanceEvent objects. It is serialized as a JSON structure by the NDEx API. The root of the tree structure is a ProvenanceEntity object representing the current state of the network. Each ProvenanceEntity may have a single ProvenanceEvent object that represents the immediately prior event that produced the ProvenanceEntity. In turn, linked to network of ProvenanceEvent and ProvenanceEntity objects representing the workflow history that produced the current state of the Network. The provenance history records significant events as Networks are copied, modified, or created, incorporating snapshots of information about “ancestor” networks.
    - Attributes in ProvenanceEntity
        - **uri** :URI of the resource described by the ProvenanceEntity. This field will not be set in some cases, such as a file upload or an algorithmic event that generates a network without a prior network as input
        - **creationEvent** : ProvenanceEvent. has semantics of PROV:wasGeneratedBy
        - **properties**: array of SimplePropertyValuePair objects
    - Attributes in ProvenanceEvent
        - **endedAtTime**: timestamp. Has semantics of PROV:endedAtTime

- **startedAtTime**: timestamp. Has semantics of PROV:endedAtTime
- **Inputs**: array of ProvenanceEntity objects. Has semantics of PROV:used.
- **properties**: array of SimplePropertyValuePair

Set Network Provenance

/network/{networkid}/provenance

Method: PUT

Authentication: Required

Description:

- Updates the Provenance aspect of the network specified by *networkid* to be the ProvenanceEntity object in the PUT data.
- The ProvenanceEntity object is intended to represent the current state and history of the network and to contain a tree-structure of ProvenanceEvent and ProvenanceEntity objects that describe the networks provenance history.
- Errors if the authenticated user does not have ADMIN permissions to the specified network.
- Errors if networkid does not correspond to an existing network.

Get Network CX Metadata Collection

/network/{networkid}/aspect?accesskey={accessKey}

Method: GET

Requires Authentication: Optional

Description:

- Returns the CX metadata collection of the network specified by *networkid*.
- Errors if *networkid* does not correspond to an existing network.

Get Network Aspect Metadata

/network/{networkid}/aspect/{aspectName}/metadata

Method: GET

Requires Authentication: Optional

Description:

- Returns the CX metadata for the specified aspect of the network specified by *networkid*.
- Errors if *networkid* does not correspond to an existing network.

Get a Network Aspect As CX

/network/{networkid}/aspect/{aspectName}?size={limit}

Method: GET

Requires Authentication: Optional

Description:

- Returns a JSON array of CX elements from the aspect specified by *aspectName* from the network specified by *networkid*.
- The *size* parameter is optional, by default the server will return all elements of this aspect.
- Errors if *networkid* does not correspond to an existing network.

Update Aspects of a Network

/network/{networkid}/aspects

Method: PUT

Authentication: Required

Description:

- Updates the aspects of the network specified by *networkid*.
- PUT data contains a CX object which includes the specified aspects that will be used to overwrite or insert into the network.
- Note: the networkAttributes aspect should not be updated using this function because it needs some special handling. Please use the Update Network Profile or Set Network Properties functions to change values in networkAttributes.

Clone a Network

/network/{networkid}/copy

Method: POST

Authentication: Required

Description: This function clones the network specified by networkid to the signed in user's account. It returns the URL of cloned network to the client.

Update Network Profile and Properties

/network/{networkid}/summary

Method: PUT

Authentication: Required

Description: This function uses the name,description, version, visibility and properties fields in the payload to overwrite the corresponding fields of the given network on server.

Get Access Key of Network

/network/{networkid}/accesskey

Method: GET

Authentication: Required

Description: This function returns an access key to the user. This access key will allow any user to have read access to this network regardless if that user has READ privilege on this network.

- The caller has to be the owner of this network.
- If the access key is not turned on, this function returns HTTP code 204 (No content).
- If an access key has been -turned on, this function returns the key.

Disable/enable Access Key on Network

/network/{networkid}/accesskey?action=disable|enable

Method: PUT

Authentication: Required

Description: This function turns off/on the access key. It returns the accessKey if the action=enable and returns nothing when action is disable. The caller has to be the owner of this network.

Network Set

Each network sets is owned by a user.

Create a Network Set

/networkset

Method: POST

Authentication: Required

Description: Create a network set. The posted object should have these 2 fields:

- name: String. A short name for the network set. Names are not unique across all users, but they should be unique within a user.
- description: String. Optional.

Update a Network Set

/networkset/{networksetid}

Method: PUT

Authentication: Required

Description:

- Updates a project based on the serialized project object in the PUT data.
- The structure of the posted project should be:

```
{ “name”: string, “description”: string}
```

Delete a Network Set

/networkset/{networksetid}

Method: DELETE

Authentication: Required

Description: Deletes a network set.

Get a Network Set

/networkset/{networksetid}?accesskey={accesskey}

Method: GET

Authentication: Not required

Description:

- Returned object has this structure:

```
{
“name”: string,
“description” : string,
“Networks”: [ network_ids]
}
```

Add networks to Network Set

/networkset/{networksetid}/members

Method: POST

Authentication: Required

Description: Add a list of networks to this set. The posted data is a list of network ids. All the networks should be visible to the owner of network set.

Delete networks from Network Set

/networkset/{networksetid}/members

Method: DELETE

Authentication: Required

Description: Delete networks from a networks set. Posted data is a list of network ids.

Update Network Set System Properties

/networkset/{networksetId}/systemproperty

Method: PUT

Authentication: Required

Description:

- Network Set System properties are the properties that describe the network set's status in a particular NDEx server.
- Sets the system property specified in the PUT data for the network set specified by networksetid.
- As of NDEx V2.0 the supported system properties are:
  - showcase: boolean. Authenticated user can use this property to control whether this network set will display in his or her home page. Caller will receive an error if the user is not the owner of the network set.
  - PUT data format: `{property: value}` such as `{ "showcase": true}`

Batch operations

Get Users By UUIDs

/batch/user

Method: POST

Authentication: Not required

Description:

- Returns a JSON array of User objects selected by the POSTed JSON array of user UUIDs.
- The number of POSTed user UUIDs is limited to 2000.

Get Groups By UUIDs

/batch/group

Method: POST

Authentication: Not required

Description:

- Returns a JSON array of Group objects selected by the POSTed JSON array of Group UUIDs.
- The number of POSTed group UUIDs is limited to 2000.

Get Network Summaries By UUIDs

/batch/network/summary?accesskey={accessKey}

Method: POST

Authentication: Optional

Description:

- Return a JSON array of network summary objects selected by the POSTed JSON array of Network UUIDs.
- This function only returns summaries for public networks is the user is not authenticated, otherwise, it returns public and the networks for which the authenticated user has READ permission.
- The optional accessKey parameter is supposed to be the accessKey of a network set. If a UUID in the POSTed list is a member of that network set, the server will bypass the permission checking on that network and include this network in the result.
- The number of POSTed Network UUIDs is limited to 2000.

Get Network Permissions By UUIDs

/batch/network/permission

Method: POST

Authentication: Required

Description:

- This function returns what permissions the authenticated user has on the given list network ids.
- Returns a JSON map in which the keys are network UUIDs and values are the highest permission assigned to the authenticated user.
- The number of POSTed network UUIDs is limited to 500.

Export Networks

/batch/network/export

Method: POST

Authentication: Required

Description:

- Creates network export tasks for the set of networks specified by the networkIds property in the network export request object in the POST data.
- The export request specified the format to export with the exportFormat property. The format is not limited to network formats; it can potentially include other formats derived

from the stored network such as gene lists in GSEA format. The **/admin/status?format=full** function can be used to get the complete list of importers/exporters that the server supports.
- Returns a JSON object which maps Network UUIDs to ExportTask IDs. The taskID can be used to download the exported file.
- The number of POSTed network UUIDs is limited to 1000.
- The structure of the network export request is:

{

“exportFormat”: “GSEA Gene Set”,

“networkIds”: a list of network UUIDs

}

Search

Search Users

/search/user?start={number}&size={number}

Method: POST

Authentication: Optional

Description:

- Returns a SearchResult object which contains an array of User objects and the total hit count of the search.
- The POSTed JSON object must have a *searchString* parameter.
- The *start* and *size* parameter are optional. The default values are *start* = 0 and *size* = 100

Search Groups

/search/group?start={number}&size={number}

Method: POST

Authentication: None

Description:

- Returns a SearchResult object which contains an array of Group objects and the total hit count of the search.
- The POSTed JSON object must have a *searchString* parameter.
- The *start* and *size* parameter are optional. The default values are *start* = 0 and *size* = 100

Search Networks

/search/network?start={number}&size={number}

Method: POST

Authentication: Optional

Description:

- Returns a SearchResult object which contains an array of NetworkSummary objects and total hit count of the search.
- The POSTed JSON object must have a *searchString* parameter.
- The *start* and *size* parameter are optional. The default values are *start* = 0 and *size* = 100
- The search can also be constrained to networks owned by a specified account, by permissions, by group permissions as specified by the following parameters in the POSTed JSON object:

searchString	Required. A whitespace-delimited string that is handled according to Lucene search string (https://lucene.apache.org/core/2_9_4/queryparsersyntax.html) protocol.
permission	Optional. By default, this parameter is not set, meaning that there is no filtering of networks based on permission. If this parameter is set, it must be either 'WRITE' or 'READ'. If set to 'WRITE', the search will only return networks for which the authenticated user has **edit or admin** permission. If set to READ, the search will only return networks for which the authenticated user has **edit, admin, or read** permission. Note that this only includes networks that are readable due to explicit permission, not networks that are readable because they have been made PUBLIC.
includeGroups	Optional. Boolean value, defaults to false. If includeGroups is true, the search will also return networks based on permissions from the authenticated user's group memberships. This enables search to return a network where a group has permission to access the network and the user is a member of the group.
accountName	Optional. String value. If the accountName parameter is provided, then the search will be constrained to networks owned by that account.

Query Network

/search/network/{networkid}/query?save={true|false}

Method: POST

Authentication: Optional

Description:

- Returns a CX network that is a 'neighborhood' subnetwork of the network specified by *networkid*.
- The query finds the subnetwork by a traversal of the network starting with nodes associated with identifiers specified in the POSTed JSON query object.
- The default value for parameter save is false. Only an authenticated user can call this function with this parameter set to true. When save=true, the server will save the query result as a network in the caller's account and return the URL of the network resource to the user. Saving query result as a network is a asynchronous call, which means the server will return the URL of the network resource (with HTTP return code 201) while the server is still running the query. User can use the returned UUID to check the status of the network creation process.

searchString	A whitespace delimited string of search terms which are matched vs. (1) the controlled vocabulary terms used in the network and (2) names of nodes in the network. A set of initial nodes is selected based on association with matched terms or simple name match. The query selects edges based on traversal from those initial nodes.
searchDepth	Integer value between 1 and 3. Sets the maximum number of traversal steps from the initial nodes. Default value is 1. If the searchDepth is less than 1, the server will do a 1 step search.
edgeLimit	Maximum number of edges that this query can return. Default value is 0. A negative or zero edgeLimit means no limit in the query.
errorWhenLimitIsOver	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**.
directOnly	Default value is false, which means full neighborhood will be returned. When this parameter is set to true, the server will perform an **Adjacent** query, which means edges between n-th step neighbor vertex will be excluded from the result.

Interconnect Query

/search/network/{networkid}/interconnectquery?save={true|false}

Method: POST

Authentication: Optional

Description:

- Returns a CX network that is a 'neighborhood' subnetwork where all the paths must start and end at one of the query nodes in the network specified by *networkid*.
- The query finds the subnetwork by a traversal of the network starting with nodes associated with identifiers specified in the POSTed JSON query object.
- The default value for parameter save is false. Only an authenticated user can call this function with this parameter set to true. When save=true, the server will save the query result as a network in the caller's account and return the URL of the network resource to the user. Saving query result as a network is a asynchronous call, which means the server will return the URL of the network resource (with HTTP return code 201) while the server is still running the query. User can use the returned UUID to check the status of the network creation process.

searchString	A whitespace delimited string of search terms which are matched vs. (1) the controlled vocabulary terms used in the network and (2) names of nodes in the network. A set of initial nodes is selected based on association with matched terms or simple name match. The query selects edges based on traversal from those initial nodes.
searchDepth	Integer value between 1 and 2. Sets the maximum number of traversal steps from the initial nodes. Default value is 1. If the searchDepth is less than 1, the server will do a 1 step search. If you want to run a **Interconnect** query described in NDEx web app, set this value to 2. If you want to run a **Direct** query, set this value to 1.
edgeLimit	Maximum number of edges that this query can return. Default value is 0. A negative or zero edgeLimit means no limit in the query.
errorWhenLimitIsOver	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**.

Search Networks by Gene/Protein

/search/network/genes?start={number}&size={number}

Method: POST

Authentication: Optional

Description:

- Returns a SearchResult object which contains an array of NetworkSummary objects and total hit count of the search.
- The POSTed JSON object must have a *searchString* parameter.
- The *start* and *size* parameter are optional. The default values are *start* = 0 and *size* = 100
- POSTed JSON object has these fields:

here is an example of the POSTed object:

```
{
"searchString": "k-ras ENGASE AMY1A BRAF"
}
```

NDEx v1.3 API methods supported for backward compatibility

Get Information about a Network by Accession (UUID)

```
GET : /network/{networkId}
```

Each network stored on an NDEx Server is assigned a universally unique identifier – a UUID. An application can query an NDEx to get summary information about the network (and determine if it is present on the NDEx) by the UUID using this method which retrieves a NetworkSummary object for the network specified by ‘networkId'. This method returns an error if the network is not found or if the authenticated user does not have READ permission for the network.

Java Client Method

public NetworkSummary getNetworkSummaryById(String networkId)

Find a Network by Search

```
POST : /network/search/{skipBlocks}/{blockSize}
```

This method returns a list of NetworkSummary objects based on a POSTed query JSON object. The maximum number of NetworkSummary objects to retrieve in the query is set by the integer value ‘blockSize' while ‘skipBlocks' specifies number of blocks that have already been read.

As of NDEx v1.3, networks are matched based on the text in designated network attributes and internal properties that are indexed as fields by a Solr engine deployed in tandem with the main NDEx database. The "v1.3 Network Query and Network Search" document describes the indexed network and node fields in detail. The search can also be constrained to networks owned by a specified account, by permissions, by group permissions as specified by the following parameters:

searchString	Required. A whitespace-delimited or comma-delimited string that each term is a gene symbol or identifier. This search function will expend each term to it’s alias, NCBI gene ID, Uniprot IDs, ensembl gene ID, and HGNC ID using gene query service at Mygene.Info.
searchString	Required. A whitespace-delimited string that is handled according to Lucene search string (https://lucene.apache.org/core/2_9_4/queryparsersyntax.html) protocol.
permission	Optional. By default, this parameter is not set, meaning that there is no filtering of networks based on permission. If this parameter is set, it must be either ‘WRITE' or ‘READ'. If set to ‘WRITE', the search will only return networks for which the authenticated user has **edit or admin** permission. If set to READ, the search will only return networks for which the authenticated user has **edit, admin, or read** permission. Note that this only includes networks that are readable due to explicit permission, not networks that are readable because they have been made PUBLIC.
includeGroups	Optional. Boolean value, defaults to false. If includeGroups is true, the search will also return networks based on permissions from the authenticated user's group memberships. This enables search to return a network where a group has permission to access the network and the user is a member of the group.
accountName	Optional. String value. If the accountName parameter is provided, then the search will be constrained to networks owned by that account.

Note that in v1.3, the "canRead" parameter documented in v1.2 is no longer supported because it was only needed for the discontinued "DISCOVERABLE" network visibility status.

Examples of search strings:

pancreatic

Simple search term. Finds networks in which any indexed term is the string "pancreatic".

panc*

Wildcarding with "*". Find networks in which any indexed field starts with "panc"

description:cancer

Search by field. Find networks in which a specific field – "description" – matches "cancer"

nodeCount:[5 TO 70]

Find networks with field nodeCount in specified range

nodeCount:[5 TO 70] AND panc*

Boolean combination of search strings using AND. Finds networks satisfying both search criteria.

Java Client Methods:

public List<NetworkSummary> findNetworks(

String searchString,

boolean canRead,

String accountName,

int skipBlocks,

int blockSize)

public List<NetworkSummary> findNetworks(

String searchString,

boolean canRead,

String accountName,

Permissions permissionOnAcc, boolean includeGroups,

int skipBlocks,

int blockSize)

Get the Provenance history for a Network

```
GET : /network/{networkId}/provenance
```

This method retrieves the ‘provenance' attribute of the network specified by ‘networkId', if it exists. The returned value is a JSON ProvenanceEntity object which in turn contains a tree-structure of ProvenanceEvent and ProvenanceEntity objects that describe the provenance history of the network. See the document NDEx Provenance History (/network-provenance-history/) for a detailed description of this structure and best practices for its use.

Java Client Method:

public ProvenanceEntity getNetworkProvenance(String networkId)

Modify the Provenance History for a Network

```
PUT : /network/{networkId}/provenance
```

Requires Authentication

Updates the ‘provenance' field of the network specified by ‘networkId' to be the ProvenanceEntity object in the PUT data. The ProvenanceEntity object is expected to represent the current state of the network and to contain a tree-structure of ProvenanceEvent and ProvenanceEntity objects that describe the networks provenance history.

Java Client Method:

public ProvenanceEntity setNetworkProvenance(

String networkId,

ProvenanceEntity provenance)

Get a Network as CX

```
GET : /network/{networkId}/asCX
```

This method retrieves all CX aspects of the entire network specified by ‘networkId' as a CX stream. This is performed as a monolithic operation, so care should be taken when requesting very large networks. Applications can use the getNetworkSummary method to check the node and edge counts for a network prior to making the request, but with the advent of CX, it is also

possible for a client to read incrementally and abort the operation if the in-memory structures become too large. As an optimization, networks that are designated read-only (see **Make a Network Read-Only**) are cached by NDEx for rapid access.

Java Client Method:

public InputStream getNetworkAsCXStream(String id)

Create a Network from CX

```
POST : /network/asCX
```

Requires Authentication

This method creates a new network on the NDEx Server based on a POSTed InputStream object. An error is returned if the stream is not provided or if the CX data does not specify a name attribute. An error is also returned if the number of elements in the CX stream is larger than a maximum size for network creation set in the NDEx server configuration. The UUID for the new network is returned.

Java Client Method:

public UUID createCXNetwork(InputStream cxStream)

Update an Entire Network as CX

```
PUT : /network/asCX
```

Requires Authentication

This method updates an existing network with new content. The method takes a network UUID and a CX Stream the PUT data. This method errors if the stream or network UUID is not provided or if the UUID does not correspond to an existing network on the NDEx Server. It also errors if the number of CX elements is larger than a maximum size for network creation set in the NDEx server configuration. The UUID corresponding to the updated network is returned.

Java Client Method:

public UUID updateNetwork(UUID networkId, InputStream cxStream)