NDEx 2.1 REST API

Last updated: June 29, 2017

Overview

The NDEx 2.1 API is a major revision of the API structure to reflect better REST practices and support the new functionalities of the NDEx 2.0 server.
All NDEx 2.1 API functions are available at the following endpoints: **http://www.ndexbio.org/v2** or **http://public.ndexbio.org/v2**

In addition, a small set of NDEx 1.3 API functions used by existing applications are still supported for backward compatibility.
NDEx 1.3 API functions are available at these endpoints: **http://www.ndexbio.org/rest** or **http://public.ndexbio.org/rest**

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 documented at: http://www.home.ndexbio.org/data-model/ (http://www.home.ndexbio.org/data-model/)

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 (https://github.com/ndexbio/ndex-python) also available via PyPi.
- Java Client (https://github.com/ndexbio/ndex-python) and Java Object Model (https://github.com/ndexbio/ndex-object-model)
- R Client (https://github.com/frankkramer-lab/ndexr) also available via Bioconductor.

API Design Guidelines Used in NDEx 2.1

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.

5 – Errors:

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

b. The body of 500 error responses are a json value that includes the error message from the server.

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

Create User

/user

Method: POST

Authentication: Not required

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.
- This is an example of the posted user object (more details of each attribute can be found in the “Get User By UUID” function):

```
{
"properties": {},
  "displayName": null,
  "isIndividual": true,
  "userName": "maxpain",
  "password": "1111111",
  "emailAddress": "maxpain@mac.com",
  "firstName": "Max",
  "lastName": "Pain",
  "image": "http://example.com/image/1.jpg",
  "website": "http://www.linkedin.com/in/maxpain",
  "description":
}
```

Delete User

/user/{userid}

Method: DELETE

Authentication: required

Description:

- Deletes the authenticated user, removing any other objects in the database that depend on the user.
- Errors – if this operation orphans a network or group

Update User

/user/{userid}

Method: PUT

Authentication: Required

Description:

- Updates the authenticated user based on the serialized user object in the PUT data.
- Errors – if the user object references a different user.
- The userName and UUID fields in the posted object are ignored by the server because they cannot be modified.

Get User By userName

/user?username={userName}

Method: GET

Authentication: Not required

Description:

- Return the user corresponding to the provided user name.
- Error if this account is not found.
- If the user account has not been verified by the user yet, the returned object will contains no user UUID and the isVerified field is false.

Authenticate a User

/user?valid=true

Method: GET

Authentication: required

Description:

- Authenticates the combination of userName and password supplied in the Basic Auth header.
- Returns the authenticated user JSON object if successful.

Get User By UUID

/user/{userid}

Method: GET

Authentication: Not required

Description:

- Returns the user JSON object corresponding to the user's UUID provided in the userid route parameter.
- The returned user object has this structure:

Attribute	Data type	Description
externalId	string	uuid of this user.
userName	string	User's account name.
password	string	Always null
isVerified	string	True if this user has verified his email address.
emailAddress	string	Email address of this user.
creationTime	Timestamp	Time when this user's account was created in NDEx
modificationTime	Timestamp	Time when this account was last modified.
isDeleted	boolean	False if this user's account is active.
isIndividual	boolean	True if this account is for an individual user. False means this account is for an organization or a project etc
displayName	string	Display name of this account, only applied to non-individual accounts.
firstName	string	Account owner's first name, only applies to individual accounts.
lastName	string	Account owner's last name, only appliies to individual accounts.
properties	object	Properties of of this user.
image	string	URL of the account owner's image.
website	string	URL of the account owner's web site
description	string	Short description of this user.

Change Password

/user/{userid}/password

Method: PUT

Authentication: required

Description:

- Changes the authenticated user's password to the new password in the PUT body.

Email New Password

/user/{userid}/password?forgot=true

Method: PUT

Authentication: Not required

Description:

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

Verify a User

/user/{userId}/verification?verificationCode={verificationCode}

Method: GET

Authentication: Not required

Description:

- Verify the given user with UUID and verificationCode.

- Returns string “User account XXX has been activated.” when this user’s account is successfully activated.

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 Network Permissions

/user/{userid}/permission?permission={permission}&start={startPage}&size={pageSize}&directonly={true|false}

Method: GET

Authentication: Optional

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

Get User’s Showcase Networks

/user/{userid}/showcase

Method: GET

Authentication: Optional

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.

Get User’s Account Page Networks

/user/{userid}/networksummary

Method: GET

Authentication: Required

Description:

- This is a convenience function designed to support “My Account” pages in NDEx applications. It returns a list of NetworkSummary objects to display.
- Userid must be the authenticated user’s UUID.

Get All Network Sets owned by a user

/user/{userid}/networksets

Method: GET

Authentication: Not required

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

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": “http://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.
- 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.

Request and Response

Create User Permission Request

/user/{userid}/permissionrequest

Method: POST

Authentication: Required

Description:

- Creates a request to ask the owner of the network for permission for access by the authenticated user.
- The type of permission and the network UUID is specified in the JSON request object.

POSTed Request Object:

{ “networkid’: UUID,

“permission”: (READ|WRITE),

“message”: string };

Create Group Permission request

/group/{groupid}/permissionrequest

Method: POST

Authentication: Required

Description:

- Create a request to ask the owner of a network for permission for access by a group.
- The authenticated user must be a member of the specified group.
- The type of permission and the network UUID are specified in the JSON request object.

POSTed Request Object:

```
{
“networkid” : UUID,
“permission” : READ|WRITE
“message” : string
}
```

Create Membership Request

/user/{userid}/membershiprequest

Method: POST

Authentication: Required

Description:

- Create a request to the group admin for the authenticated user to join the specified group.
- The type of membership and the group UUID are specified in the JSON request object.

POSTed Request Object:

```
{
“groupid”:  UUID
“type”:  MEMBER|GROUPADMIN,
“message”: string
}
```

Get a User’s Permission Requests

/user/{userid}/permissionrequest?type={sent|received}

Method: GET

Authentication: Required

Description:

- Returns a JSON array of permission request objects in which the authenticated user is either the recipient or the sender.
- The type parameter filters the request object for either sent or received. This function returns both type of requests if type parameter is omitted.

Get a User’s Permission Requests by id

/user/{userid}/permissionrequest/{requestid}

Method: GET

Authentication: Required

Description:

- Returns the permission request object specified by requestid.
- The authenticated user must be either the sender or recipient of the request.

Get a Group’s Permission Requests

/group/{groupid}/permissionrequest?networkid={networkid}&permission={permissionType}

Method: GET

Authentication: Required

Description:

- Returns a JSON array of permission request objects for the specified network sent to the specified group.

Get a Group’s Permission Requests by id

/group/{groupid}/permissionrequest/{requestid}

Method: GET

Authentication: Required

Description:

- Returns the permission request object specified by requestid sent to the specified group.
- The authenticated user must be either a member of admin of the group.
- A Request object has these attributes:

Attribute	Data type	Description
externalId	string	uuid of this request.
creationTime	Timestamp	Time when this request was created in NDEx
modificationTime	Timestamp	Time when this request was last modified.
requesterId	String	UUID of the user who creates this request.
requestType	string	Type of the request. Can be one of these values: “UserNetworkAccess”, “JoinGroup”, “GroupNetworkAccess”
permission	string	Type of permission this request is asking for. Can be one of these values: “READ”, “WRITE”
message	string	A short message send together with this request.
destinationUUID	string	The UUID of the requested resource.
sourceUUID	string	Source UUID of this request.
message	string	Error message if this task failed.
resource	uuid	UUID of the resource which this task operates on.
destinationName	string	Name of the destination resource
sourceName	string	Name of the source resource.
responder	string	UserName of the user who responded to this request
response	string	Type of response. Can be one of these values: “DECLINED”, “ACCEPTED”, or “PENDING”
responseTime	Timestamp	Time when the responder responded to this request.
responseMessage	string	A short message sent together with the response

Get a User’s Membership Requests

/user/{userid}/membershiprequest?type={sent|received}

Method: GET

Authentication: Required

Description:

- Returns a JSON array of membership request objects either sent or received by the authenticated user.
- Userid is the uuid of the authenticated user.

Get a User’s Membership Request by id

/user/{userid}/membershiprequest/{requestid}

Method: GET

Authentication: Required

Description:

- Returns the membership request object specified by requestid
- The membership request object must be either sent or received by the authenticated user. Userid should be the uuid of the authenticated user.

Accept or Deny a permission Request

/user/{recipient_id}/permissionrequest/{requestid}?action={accept|deny}&message={message}

Method: PUT

Authentication: Required

Description:

- Accept or deny the permission request specified by requestid.
- The authenticated user must be the recipient of the permission request

Accept or Deny a membership Request

/user/{recipient_uuid}/membershiprequest/{requestid}?action={accept|deny}&message={message"

Method: PUT

Authentication: Required

Description:

- Accept or deny the membership request specified by requestid.
- The authenticated user must be a group admin for the group specified by the request.

Delete a Permission Request

/user/{sender_id}/permissionrequest/{requestid}

Method: DELETE

Authentication: Required

Description:

- Delete the permission request specified by requestid
- The authenticated user must be the sender of the permission request

Delete a Membership Request

/user/{sender_id}/membershiprequest/{requestid}

Method: DELETE

Authentication: Required

Description:

- Delete the membership request specified by requestid
- The authenticated user must be the sender of the membership request

Network

Create a CX Network

/network

Method: POST

Authentication: Required

HTTP header: content-type: application/json

Description:

- Create a network from CX data.
- The input cx data is expected to be in the CXNetworkStream field of posted multipart/form-data.

Update a Network

/network/{networkid}

Method: PUT

Authentication: required

Description:

- Update the specified network with new content, based on CX data.
- The CX data is in the CXNetworkStream field of PUTed multipart/form-data.
- 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.

- 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 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.
- PUT data format: {property: value} such as { "readOnly": true} or {"visibility": "PUBLIC"} or a JSON object with both properties

Get All Permissions on a Network

/network/{networkid}/permission?type={user|group}&permission={permission}&start={startPage}&size={pageSize}

Method: GET

Authentication: Required

Description:

- Returns a JSON object describing the user or group permissions for the network specified by *networkid*.
- In the returned object, the keys are user or group UUIDs and the values are the highest permission assigned to that user or group.
- The *type* parameter is required and can be either 'user' or 'group'. It specifies whether user or group permissions should be returned.
- The *permission* parameter constrains the type of the returned Membership objects and may take the following set of values: READ, WRITE, and ADMIN.
  - READ, WRITE, and ADMIN are mutually exclusive.
- If the *permission* parameter is missing, all permission types will be returned.
- Errors if the specified network is not owned by authenticated user.
- Errors if networkid does not correspond to an existing network.

Update Network Permission

/network/{networkid}/permission?(userid={uuid}|groupid={uuid})&permission={permission}

Method: PUT

Authentication: Required

Description:

- Updates the permission of a user specified by userid or group specified by groupid for the network specified by *networkid*.
- The permission is updated to the value specified in the *permission* parameter.
- 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.
- Errors if it would leave the network without any user having ADMIN permissions: NDEx does not permit networks to become 'orphans' without any owner.

Delete Network Permission

/network/{networkid}/permission?(userid={uuid}|groupid={uuid})

Method: DELETE

Authentication: Required

Description:

- Removes any permission for the network specified by *networkid* for the user or group specified by memberid parameter.
- Errors if the authenticated user making the request does not have ADMIN permissions to the specified network.
- Errors if networkid does not correspond to an existing network.

- Errors if it would leave the network without any user having ADMIN permissions: NDEx does not permit networks to become 'orphans' without any owner.

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 an Aspect of a Network

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

Method: PUT

Authentication: Required

Description:

- Updates the aspect specified by *aspectName* from the network specified by *networkid*.
- PUT data contains a CX object which includes the specified aspect.

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 turn 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. User can use the get network access key function to turn the key on again.

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 use the name,description, version, visibility and properties fields in the payload to overwrite the corresponding fields of the given network on server.

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.

Get Access key of Network Set

/networkset/{networksetid}/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 member networks of this network set regardless if that user has READ privilege on that network

- The caller has to be the owner of this network set.
- If an access key was not enabled it reurns http code 204.
- If an access key has been turned on, this function returns the key.

Disable/Enable Access Key on Network Set

/networkset/{networksetid}/accesskey?action={disable|enable}

Method: PUT

Authentication: Required

Description: This function turns on/off the access key. It returns the key when it is enabled, and returns http code 204 when it is disabled.

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.

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.

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 number of POSTed Network UUIDs is limited to 2000.

Get Tasks by UUIDs

/batch/task

Method: POST

Authentication: Required

Description:

- Return a JSON array of Task objects selected by the POSTed JSON array of Task UUIDs.

Get Aspects of a Network

/batch/network/{networkid}/aspect

Method: POST

Authentication: Required

Description:

- Returns a CX document containing selected aspects of the Network specified by *networkid*.
- The aspects to return are specified in the POSTed JSON array of aspect names.

Update Aspects of a Network

/batch/network/{networkid}/aspect

Method: PUT

Authentication: Required

Description:

- Updates aspects of the network specified by *networkid*.
- The aspects to update and their new values are specified by the CX document in the PUT data – the Update CX.
- For each aspect in the Update CX document, the elements in that aspect completely replace the elements in the corresponding aspect in the updated network.

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 structure of the network export request is:

{

“exportFormat”: “cx”,

“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

Method: POST

Requires Authentication: false

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.

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.

edgeLimit	Maximum number of edges that this query can return. Default value is 1500. Server will return an error if the number of edges in the result is larger than the edgeLimit parameter.

Advanced Query

/search/network/{networkId}/advancedquery

Method: POST

Requires Authentication: false

Description:

- This method retrieves a filtered subnetwork of the network specified by 'networkId' based on a POSTed JSON query object.
- The returned subnetwork contains edges which satisfy both the edgeFilter and the nodeFilter up to a specified limit.
- The subnetwork is returned as a Network object containing the selected edges plus all other network elements relevant to the edges.

edgeFilter	The query will select edges which have any property that satisfies one or more of the propertySpecifications of the edgeFilter. One reserved property name is handled specially: **"ndex:predicate"**. For this property, the value in the propertySpecification is matched vs. the name of the predicate (relationship type) assigned to the edge. This enables the important case in which edges are filtered based on their relationship.
nodeFilter	The query will select edges which connect nodes satisfying the nodeFilter. The 'mode' attribute of the nodeFilter controls whether the filter is applied to the source node, target node, both, or either. An edge satisfies the nodeFilter if: **mode = 'Source'** and the source node has properties satisfying any propertySpecification in the list. **mode = 'Target'** and the target node has properties satisfying any propertySpecification in the list. **mode = 'Both'** and both source and target nodes have properties satisfying any PropertySpecification in the list. **mode = 'Either'** and either source and target nodes have properties satisfying any PropertySpecification in the list. Three reserved property names are handled specially: **"ndex:nodeName"**: the value in the propertySpecification is matched vs. the name of the node. **"ndex:nameOrTermName"**: the value in the propertySpecification is matched vs. either the name of the node or the name of a controlled vocabulary term that the node represents. **"ndex:functionTermType"**: the value in the propertySpecification is matched vs. the name of the controlled vocabulary term that is the function of the FunctionTerm that the node represents. This effectively enables filtering on the type of the node for OpenBEL format networks and others that employ FunctionTerms.
edgeLimit	Integer value. The query terminates and returns an error when the number of edges found exceeds this limit. When edgeLimit is set to 0 or to a negative integer, there is no limit, all edges that satisfy the query criteria will be returned.

The query is only valid if at least one filter is not null and non-empty. An error will be returned if both the nodeFilter and edgeFilter attributes are nulls or have no property specifications.

Each propertySpecification in a filter list is a property value pair in the following format:

```
{
"name" : ,
"value" :
}
```

All matches of node or edge properties vs. propertySpecifications are case-insensitive.

Query JSON:

```
{
“nodeFilter”: {
“propertySpecifications” : [ ],
“mode” :
},
“edgeFilter: {
“propertySpecifications” : [ ]
},
“edgeLimit” : ,
“queryName" : "My query”
}
```

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)