NDEx (www.ndexbio.org) is an online commons for biological networks. Users can upload, share, and distribute networks of many types, sizes, and formats. Developers can access NDEx via a web-based programming interface. NDEx promotes the publication of networks as dynamic, actionable data and the development of applications using networks.

In Brief

NDEx (www.ndexbio.org) is an online commons where scientists can upload, share, and publicly distribute biological networks of many types, sizes, and formats. It promotes the publication of networks as dynamic, actionable data and the development of applications using networks.

Authors

Dexter Pratt, Jing Chen, David Welker, ..., Jan Kuentzer, Barry Demchak, Trey Ideker

Correspondence
depratt@ucsd.edu

Highlights

- NDEx (www.ndexbio.org) is an online commons for biological networks
- Users can upload, share, and distribute networks of many types, sizes, and formats
- Developers can access NDEx via a web-based programming interface
- NDEx promotes the publication of networks as dynamic, actionable data
NDEx, the Network Data Exchange

Dexter Pratt,1,*, Jing Chen,1 David Welker,1 Ricardo Rivas,1 Rudolf Pillich,1 Vladimir Rynkov,1 Keiichiro Ono,1 Carol Miello,2 Lyndon Hicks,3 Sandor Szalma,4 Aleksandar Stojmirovic,5 Radu Dobrin,5 Michael Braxenthaler,5 Jan Kuentzer,7 Barry Demchak,1 and Trey Ideker1,8
1Department of Medicine, University of California San Diego, La Jolla, CA 92093, USA
2Pfizer Inc., Eastern Point Road, Groton, CT 06340, USA
3Pfizer Inc., 1 Burtt Road, Andover, MA 01810, USA
4Janssen Research and Development LLC, 3210 Merryfield Row, San Diego, CA 92121, USA
5Janssen Research and Development LLC, 1400 McKean Road, Spring House, PA 19477, USA
6Roche, Pharma Research and Early Development Informatics, Roche Innovation Center, New York, NY, 10016, USA
7Roche, Pharma Research and Early Development Informatics, Roche Innovation Center, Penzberg 82377, Germany
8Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA 92093, USA
*Correspondence: depratt@ucsd.edu
http://dx.doi.org/10.1016/j.cels.2015.10.001

SUMMARY

Networks are a powerful and flexible methodology for expressing biological knowledge for computation and communication. Network-encoded information can include systematic screens for molecular interactions, biological relationships curated from literature, and outputs from analyses of Big Data. NDEx, the Network Data Exchange (www.ndexbio.org), is an online commons where scientists can upload, share, and publicly distribute networks. Networks in NDEx receive globally unique accession IDs and can be stored for private use, shared in pre-publication collaboration, or released for public access. Standard and novel data formats are accommodated in a flexible storage model. Organizations can use NDEx as a distribution channel for networks they generate or curate. Developers of bioinformatic applications can store and query NDEx networks via a common programmatic interface. NDEx helps expand the role of networks in scientific discourse and facilitates the integration of networks as data in publications. It is a step toward an ecosystem in which networks bearing data, hypotheses, and findings flow easily between scientists.

Networks are a precise and computable form in which biologists can express many kinds of information, including models of biological mechanisms, experimental facts, and relationships derived by systematic data analysis. When pathway diagrams evolved into repositories of small pathway networks (Croft et al., 2014; Kanehisa and Goto, 2000; Ogata et al., 1999), they became searchable resources and the basis for data interpretation and collaborative pathway editing applications (van Iersel et al., 2008). The emergence of repositories of large networks of molecular relationships (Franceschini et al., 2013; Orchard et al., 2014; Stark et al., 2011; Warde-Farley et al., 2010) in both simple and complex formats (Demir et al., 2010; Le Novère et al., 2006; Mi et al., 2010; OpenBEL, 2011) condensed collections of data into structured findings useful for hypothesis generation and computational prediction (Bandyopadhyay et al., 2006; QIAGEN, 2015; Vandin et al., 2012). In recent years, there has been rapid progress in the construction of networks inferred by the systematic processing of genome-scale information, providing an important avenue for interpretation and a counterpoint to literature curation (Califano et al., 2012; Chuang et al., 2007; Hofree et al., 2013). By providing a flexible computable medium for biological knowledge, networks are also becoming a critical element for new models of scientific publication in which data and their derivatives are as important as text (CyNetShare, 2014).

NDEx, the Network Data Exchange, is an open-source (Data S1, Open Source) software framework that facilitates the sharing of networks of many types and formats, the publication of networks as data, and the use of networks in modular software. In comparison with repositories such as IntAct (Orchard et al., 2014) or the Kyoto Encyclopedia of Genes and Genomes (KEGG) (Kanehisa and Goto, 2000), where network content in specific formats is managed by the organization maintaining the resource, NDEx is a data commons where users manage the sharing and publication of their own networks (Data S1, Related Resources). It accommodates networks of any type, from pathway models and interaction maps to novel data-driven knowledge, handling diverse formats including simple interaction format (SIF), extensible graph markup and modeling language (XGML), BioPAX3 and OpenBEL (Data S1, Network Formats). It promotes scientific publication and reproducibility by enabling the tracking of accession and provenance of networks. Finally, NDEx provides a flexible, programmatically accessible storage service that promotes modular software development and workflows in which networks that are output from one application can be input to another. NDEx does not perform biological analysis and visualization itself but, instead, enables the interchange of networks between applications that do.

As a data commons, NDEx enables scientists and organizations that create accounts on the NDEx server to upload and save networks and to create communities of users, much like Google+ Circles or LinkedIn Groups. They can manage access to their networks, making them private, public, or shared with selected users and community groups, similar to shared document systems such as Google Docs or DropBox (Figure 1A; Data S1, NDEx Basics). The shared networks preserve the
distinct semantics of their original formats while standardizing the treatment of identifiers, citations, properties, and network topology (Figure 1B; Data S1, Data Model). NDEx therefore differs in approach from WikiPathways (Pico et al., 2008), a pioneering collaborative platform for the curation of biological pathways, where all documents are edited publically and use a single format. Organizations that publish network content can use NDEx as a channel for distribution. Networks from the NCI Pathway Interaction Database (Schaefer et al., 2009), Pathway Commons (Cerami et al., 2011), and the OpenBEL Consortium are among those available in NDEx (Data S1, Metrics).

To support the publication of networks as data, it must be possible to unambiguously specify the identity of the network and trust that the content of a published network will remain constant. NDEx provides accession identifiers for every network, assigning a universal unique identifier (UUID) that distinguishes it from all other networks across all servers. The owner of an NDEx network can set its status to be read-only, preventing further edits. These features enable networks to be reliable, consistent references suitable as inputs to further research.

When networks expressing data, hypotheses, and findings are both inputs and outputs of analysis and are referenced in publications, it becomes important to know how and when a network was created and which inputs and algorithms would be required to reproduce it. NDEx addresses these needs by including the “provenance history” (Figure 1C; Data S1, Provenance) with each network. The provenance history captures the workflow leading to the current network by describing prior events, networks, and other resources. The history grows as networks are created, modified, used, or copied. It incorporates concepts and vocabulary from ongoing work in provenance annotation (Ciccarese et al., 2013; DublinCore, 2012; PROV-O, 2013) but adopts a strategy where referenced resources are described rather than simply linked, preserving information in cases where the resources are later removed, altered, or become unavailable.

NDEx promotes the development of new network analysis algorithms and applications by expanding access to networks as inputs by facilitating immediate sharing of network results and by providing a path to publication. An NDEx server stores and manages networks and all related information and can be accessed by applications via a web-based relational state transfer application programming interface (REST API) (Fielding and Taylor, 2002). NDEx client libraries for this API have been created in the Java, Python, and R languages to facilitate easy use by scientists, although NDEx can be accessed via any language capable of internet communications (Data S1, REST API).
NDEx website (www.ndexbio.org) is the most comprehensive example of an NDEx-enabled application that accesses the public NDEx server via the REST API. The website enables visitors to anonymously search, browse, and query networks and logged-in users to upload content, manage groups, and share networks. Simple analysis scripts (e.g., Python or R) can also query NDEx via the API to obtain input networks and then save analysis result networks directly to NDEx (Figure 2). This storage service model enables the researcher to focus on the core data analysis or algorithm rather than on the management, storage, and publication of networks.

The rich Cytoscape biological analysis and visualization environment (Shannon et al., 2003) can also access NDEx via the REST API, enabling Cytoscape users to search, import, and export networks. Under Cytoscape, a workflow might start by importing a transcriptional regulatory network from NDEx, after which the user could annotate the network with a differential mRNA expression dataset and process it to find subnetworks enriched for genes with significant changes in mRNA expression. The user could then export the subnetworks back to NDEx for review by collaborators or for use as input to further analyses. The CyNDEx App (Data S1, CyNDEx) implements access to NDEx, and upcoming releases of Cytoscape are expected to incorporate its functionality into the main application, making NDEx networks immediately available to users.

An important use of NDEx is to enable new models of scientific publication via network visualization applications in which live data structures replace static diagrams and supplemental files. Readers can immediately act on networks published via NDEx as data that can be dynamically visualized, inspected, and manipulated. For example, a biologist might save selected portions of a published network to their NDEx account to capture a mechanism of interest. Both the original and saved networks would be accessible to other NDEx-capable applications for analysis and visualization. This integration of viewing, annotation, sharing, and action can accelerate and enrich the process of scientific communication.

Finally, users can download and deploy the NDEx server software for private uses that would be impractical or unsupported on the shared public server. An NDEx server can be installed behind a firewall to handle cases where strong security is required, enabling storage of proprietary networks developed for the health sciences industry or those that incorporate patient information subject to privacy standards (e.g., the Health Insurance Portability and Accountability Act [HIPAA]). A private NDEx can also be deployed on local servers or on a scientist's desktop for applications that store very large networks or perform frequent, large transactions. Applications can

Figure 2. NDEx Workflow
Example workflow in which network 1 is created by systematic analysis of genome scale data and stored in NDEx, network 2 is produced by a Cytoscape analysis that takes network 1 as an input, network 3 represents a canonical pathway uploaded from literature, and network 4 is the output of a bioinformatic script that operates on networks 2 and 3. Network 4 is made public and read-only and becomes part of a publication. Network 4 is viewed by readers of the publication using an NDEx-capable web application that enables them to directly act on the network data, such as saving a private copy to an NDEx account as network 5.
simultaneously access both public and private NDEX servers, or users can coordinate private NDEX instances with a public server using NDEX Sync (Data S1, NDEX Sync), a command line utility that can copy and update selected networks between servers.

In summary, NDEX provides distinctive capabilities as a data commons to further the use of biological networks in scientific discourse. It promotes the development of modular applications and the re-use of research products by creating a network exchange where the outputs from one project or application can readily become inputs to another. The NDEX platform is enabling new forms of publication and collaboration in which network information can be immediately analyzed, visualized, annotated, and shared.

SUPPLEMENTAL INFORMATION

Supplemental Information includes NDEX Data Exchange information and can be found with this article online at http://dx.doi.org/10.1016/j.cels.2015.10.001.

AUTHOR CONTRIBUTIONS


ACKNOWLEDGMENTS

NDEX is supported in part by F. Hoffmann-La Roche Ltd, Janssen Research and Development, LLC, Pfizer, Inc. C.M. and L.H. are employees of Pfizer Inc. S.S., A.S., and R.D. are employees of Janssen Research and Development, LLC. M.B. and N.D. are employees of F. Hoffmann-La Roche Ltd. The industry funding for NDEX is managed as a project of the Cytoscape Consortium, a non-profit corporation. T.I. is a board member and officer of the Cytoscape Consortium. B.D. is an officer of the Cytoscape Consortium. NDEX is also supported in part by the National Cancer Institute under award U24 CA-184427.

Received: March 19, 2015
Revised: September 14, 2015
Accepted: October 6, 2015
Published: October 28, 2015

REFERENCES


Cell Systems 1, 302–305, October 28, 2015 ©2015 Elsevier Inc. 305
Cell Systems
Supplemental Information

NDEx, the Network Data Exchange

Dexter Pratt, Jing Chen, David Welker, Ricardo Rivas, Rudolf Pillich, Vladimir Rynkov, Keiichiro Ono, Carol Miello, Lyndon Hicks, Sandor Szalma, Aleksandar Stojmirovic, Radu Dobrin, Michael Braxenthaler, Jan Kuentzer, Barry Demchak, and Trey Ideker
This document provides supplemental information about the version of NDEx described in the paper, the v1.2 release on July 2015. It includes discussions of the network data model, import-export formats, tracking of provenance, and the NDEx server API. It also provides user documentation for the NDEx web user interface and two associated utilities. Apart from sections 2 and 6, all materials are derived from the NDEx online documentation (www.ndexbio.org). Because the online documentation will be revised and expanded over time, we recommend that you consult that resource for the latest information on NDEx.

<table>
<thead>
<tr>
<th>NDEx, the Network Data Exchange: Supplemental Materials</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Open Source</td>
<td>2</td>
</tr>
<tr>
<td>2. Related Resources</td>
<td>4</td>
</tr>
<tr>
<td>3. Network Formats</td>
<td>10</td>
</tr>
<tr>
<td>4. NDEx Basics</td>
<td>24</td>
</tr>
<tr>
<td>5. Data Model</td>
<td>41</td>
</tr>
<tr>
<td>6. Metrics</td>
<td>51</td>
</tr>
<tr>
<td>7. Provenance</td>
<td>52</td>
</tr>
<tr>
<td>8. REST API</td>
<td>59</td>
</tr>
<tr>
<td>9. CyNDEx</td>
<td>71</td>
</tr>
<tr>
<td>10. NDEx Sync</td>
<td>78</td>
</tr>
</tbody>
</table>
1. Open Source

NDEx License

All NDEx software included in the NDEx Sources section below is available under the following BSD license:

Copyright (c) 2013, 2015, The Regents of the University of California, The Cytoscape Consortium All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NDEx Sources

All NDEx Sources are stored on GitHub in publicly accessible repositories under the ndexbio organization:https://github.com/ndexbio

The following repositories support released NDEx software. For all other repositories under ndexbio, we advise you to consult with the NDEx team on their status before using. Some are in active development while others may be obsolete or highly experimental.

NDEx Server

- https://github.com/ndexbio/ndex-rest
• https://github.com/ndexbio/ndex-object-model
• https://github.com/ndexbio/ndex-common

NDEx Web App
• https://github.com/ndexbio/ndex-webapp

NDEx Java Client
NDEx Java Client is not an application; it is a library available for use by developers to create Java applications that access NDEx.

• https://github.com/ndexbio/ndex-java-client
• https://github.com/ndexbio/ndex-object-model

CyNDEx Cytoscape App
• https://github.com/ndexbio/ndex-cytoscape-app
• https://github.com/ndexbio/ndex-java-client
• https://github.com/ndexbio/ndex-object-model

NDEx Sync Copier Utility
• https://github.com/ndexbio/ndex-sync
• https://github.com/ndexbio/ndex-java-client
• https://github.com/ndexbio/ndex-object-model
2. Related Resources

Comparison of NDEx to Other Network Resources

NDEx, the Network Data Exchange is an online resource to enable collaboration and publication using biological networks. It is a “commons”, a scientist-driven data exchange where both individuals and organizations can share networks of any type, from pathway models and interaction maps in standard formats to novel data-driven knowledge. Further, NDEx is infrastructure supporting data publication and application development by accessioning networks and presenting an API where they can be searched and accessed in a reproducible manner.

We believe that this focus differentiates NDEx from the array of biological network resources currently available to biologists. In many cases NDEx is complementary to the missions of existing network resources, potentially playing roles as a novel distribution channel, a user content management component, or a source for staging of pre-publication content.

This document summarizes 42 biological network resources in comparison to NDEx. Most of these resources can be categorized either as repositories of network structured information, as analysis applications that operate on input data (such as gene lists) via techniques that use one or more reference networks, or as both.

“Repositories” in this context, means resources where the network content is managed by the organization maintaining the resource, and is therefore different from the structure of NDEx in which the users manage the network content. Some well-known examples of repositories include KEGG (http://www.genome.jp/kegg/pathway.html), Pathway Commons (http://www.pathwaycommons.org/about/), IntAct (http://www.ebi.ac.uk/intact/), and BioCyc (http://biocyc.org/).

Many repositories also differ from NDEx because they use specific network formats and models of biology, in contrast to the NDEx strategy of supporting many formats in a common framework. NDEx provides a novel distribution strategy for organizations that maintain repositories, a new channel for their content to reach users and applications.

Analysis applications using network resources include sites such as GeneMania (http://www.genemania.org/) and NCI DAVID (https://david.ncifcrf.gov/). Although NDEx provides some search and query operations that could be construed as “analysis”, its mission is not to perform biological analyses but instead to be a service that facilitates the creation of applications, both as a source of reference networks and as a place for users to store network-structured analysis results. A recent example of a network-oriented analysis application is Network Portal by the Institute for Systems Biology (http://networks.systemsbiology.net/), which “provides analysis and visualization tools for selected gene regulatory networks to aid researchers in biological discovery and hypothesis development.” Its design includes several features to promote data sharing and integration with other applications, but its primary focus is analysis, using networks of transcriptional regulation, distinct from the NDEx mission.

WikiPathways (http://www.wikipathways.org/index.php/WikiPathways) is a pioneering collaborative platform for the curation of biological pathways, a resource that shares the NDEx goal of facilitating scientific discourse by providing a platform for user-driven content. It differs, however, in that (1) it is focused on pathway diagrams that are small, curated, and in which the content may not be fully represented as a network and (2) it employs the “Wiki” model of collaboration on a public document, different from the “Google Docs” approach of NDEx in which users manage the access to their networks. The role of NDEx in the context of
collaborative environments such as WikiPathways could be as a “back end” resource to store and share the content created by the collaborators.

BioModels at EBI (https://www.ebi.ac.uk/biomodels-main/) is an example of a database of biological information that could be considered a network resource, but which is different from NDEex not only because the content is managed but also because it is specialized to a particular kind of biological data structure. BioModels is a “repository of computational models of biological processes”, serving as resource for the computational modeling community. Although there are forms of these computational models that can be expressed as networks (and which NDEx may support at some point), BioModels presents these models in a comprehensive manner tailored to the needs of its user community.

**Resources**

The following sections present a list of network repositories and selected examples of network-oriented analysis applications. The repositories include both those that are based on curated mechanistic information (“pathways”) and those that are focused on interaction data. (Special thanks to the maintainers of PathGuide http://www.pathguide.org/. PathGuide was an invaluable resource in preparing this document.)

**Aggregators of Network Resources**

Pathway Commons - http://www.pathwaycommons.org/about/
- “Pathway Commons is a network biology resource and acts as a convenient point of access to biological pathway information collected from public pathway databases, which you can search, visualize and download.”
- Aggregator of network repository data from many sources
- Normalizes resources to BioPAX3
- Distributes in SIF and BioPAX3 formats

iRefIndex - http://irefindex.org/wiki/index.php?title=iRefIndex
- “Provides an index of protein interactions available in a number of primary interaction databases including BIND, BioGRID, CORUM, DIP, HPRD, InnateDB, IntAct, MatrixDB, MINT, MPact, MPIDB, MPPI and OPHID.”

**Protein-Protein and Other Molecular Interaction Networks**

BIND Biomolecular Interaction Network Database
- No longer maintained, content incorporated in several other repositories

BioGRID - http://thebiogrid.org/
- “BioGRID is an interaction repository with data compiled through comprehensive curation efforts.”

CCSB Interactome - http://interactome.dfci.harvard.edu/
- A repository of experimentally derived protein interactions

DIP Database of Interacting Proteins - http://dip.doe-mbi.ucla.edu/dip/Main.cgi
- “The DIP™ database catalogs experimentally determined interactions between proteins.”

IntAct molecular interaction database - http://www.ebi.ac.uk/intact/
- A central, standards-compliant repository of molecular interactions, including protein–protein, protein–small molecule and protein–nucleic acid interactions.
- IntAct provides both an open source database system and analysis tools for molecular interaction data.

- “NetPro™ is a comprehensive database of Protein-Protein and Protein-Small molecules interaction, consisting of more than 320,000 interactions captured from more than 1500 abstracts, approximately 1600 published journals and more than 60,000 references.”

STRING - http://string-db.org/
- “STRING is a database of known and predicted protein interactions. The interactions include direct (physical) and indirect (functional) associations.”

MINT: Molecular INTeraction database - http://mint.bio.uniroma2.it/mint/Welcome.do
- “MINT focuses on experimentally verified protein-protein interactions mined from the scientific literature by expert curators.”
- Now integrated with IntAct.

RNA-binding protein database RBPDB - http://rbpdb.ccbr.utoronto.ca/
- Repository of RNA-protein interactions.

BioLiP - http://zhanglab.ccmb.med.umich.edu/BioLiP/
- “BioLiP is a semi-manually curated database for high-quality, biologically relevant ligand-protein binding interactions.”

BindingDB - http://www.bindingdb.org/bind/index.jsp
- “BindingDB is a public, web-accessible database of measured binding affinities, focusing chiefly on the interactions of protein considered to be drug-targets with small, drug-like molecules.”

- Commercial repository of gene regulation interactions, subset available for academic use.

iMEX - http://www.imexconsortium.org/
• “A non-redundant set of protein-protein interaction data from a broad taxonomic range of organisms”

Protein interaction data curation: the International Molecular Exchange (IMEx) consortium Nat Methods 2012, 9, 345-350

TAP Project - http://tap.med.utoronto.ca/exttap/
• “The Yeast TAP Project is aimed at elucidating the entire network of protein-protein interactions in a model eukaryotic organism, namely the yeast Saccharomyces cerevisiae.”
• Repository derived from experimental data using tandem affinity purification (TAP).

Pathway Network Resources

Netpath - http://www.netpath.org/
• “NetPath’ is a manually curated resource of signal transduction pathways in humans.”

• “Biomolecular interactions and cellular processes assembled into authoritative human signaling pathways”
• Cancer focused
• Soon to use NDEx and Pathway Commons as its primary distribution mechanisms, will no longer be updated.

Reactome - http://www.reactome.org/
• “Reactome is a free, open-source, curated and peer reviewed pathway database. Our goal is to provide intuitive bioinformatics tools for the visualization, interpretation and analysis of pathway knowledge to support basic research, genome analysis, modeling, systems biology and education.”
• Includes analysis tools.

SignaLink Database - http://signalink.org/
• “SignaLink 2.0: An integrated resource to analyze signaling pathway cross-talks, transcription factors, miRNAs and regulatory enzymes”
• Includes analysis tools.

• “WikiPathways is an open, public platform dedicated to the curation of biological pathways by and for the scientific community.”
• User submitted content. Wiki model, related but not identical to NDEx “exchange” model.
• Pathway diagrams are sometimes only partially computable, incorporating graphic elements with meaning apparent to the biologist but difficult for algorithms to interpret.

BioCyc Database Collection - http://biocyc.org/
• “BioCyc is a collection of 5711 Pathway/Genome Databases (PGDBs), plus software tools for understanding their data.”
• Includes EcoCyc and MetaCyc.
• Repository of pathway networks with a focus on metabolism.

• Repository of pathway networks and interactions
• Manual curation of both relationships and diagrams

MANET database - [http://manet.illinois.edu/aboutManet.php](http://manet.illinois.edu/aboutManet.php)
• “The Molecular Ancestry Network (MANET) database project traces evolution of protein architecture onto biomolecular networks.”

Small Molecule Pathway Database (SMPDB) - [http://smpdb.ca/](http://smpdb.ca/)
• “An interactive, visual database containing more than 618 small molecule pathways found in humans.”
• Extensive, carefully formatted diagrams
• Exports in BioPAX3 and SBGN

Atlas of Cancer Signaling Networks - [https://acsn.curie.fr](https://acsn.curie.fr)
• “ACSN is a pathway database and a web-based environment that contains a collection of interconnected cancer-related signaling network maps”
• Unique graphic interface
• Uses SBGN created with Cell Designer

• “The UCSD Signaling Gateway Molecule Pages provide essential information on over thousands of proteins involved in cellular signaling.”
• Includes links to pathways in several repositories.

SPIKE - [http://www.cs.tau.ac.il/~spike/](http://www.cs.tau.ac.il/~spike/)
• “SPIKE is a database of highly curated human signaling pathways with an associated interactive software tool.”
• Incorporates information from other repositories in the curation process.

BIGG - [http://bigg.ucsd.edu/](http://bigg.ucsd.edu/)
• “BiGG is a knowledgebase of Biochemically, Genetically and Genomically structured genome-scale metabolic network reconstructions”

HumanNet - [http://www.functionalnet.org/humannet/](http://www.functionalnet.org/humannet/)
• “A probabilistic functional gene network of 18,714 validated protein-encoding genes of Homo sapiens (by NCBI March 2007), constructed by a modified Bayesian integration of 21 types of ‘omics’ data from multiple organisms, with each data type weighted according to how well it links genes that are known to function together in H. sapiens.”

• Large proprietary database of molecular interactions integrated with analysis tools

• Large proprietary database
• “Manually curated database of mammalian biology and medicinal chemistry data”

• Large proprietary database integrated with analysis tools

Related Biological Repositories

BioModels - https://www.ebi.ac.uk/biomodels-main/
• BioModels Database is a “repository of computational models of biological processes”.
• Models described from literature are manually curated and enriched with cross-references.

The Cell Collective - http://thecellcollective.org
• Virtual cell models for simulations
• Related to NDEx in that they also support a “crowdsourcing” strategy.

• Curated pathway diagrams
• Not a network resource – only diagrams and gene lists are available, no computable connectivity.

Selected Examples of Network-Oriented Analysis

GeneMania - http://www.genemania.org/
• “GeneMANIA finds other genes that are related to a set of input genes, using a very large set of functional association data.”

• “Provides analysis and visualization tools for selected gene regulatory networks to aid researchers in biological discovery and hypothesis development.”

DAVID - https://david.ncifcrf.gov/
• Gene set analysis enrichment scoring includes pathways.

MSigDB - http://www.broadinstitute.org/gsea/msigdb/index.jsp
• Gene set analysis enrichment scoring includes pathways.

GenomeSpace - http://www.genomespace.org
• “GenomeSpace is a cloud-based interoperability framework to support integrative genomics analysis through an easy-to-use Web interface.”
• Integration includes network-oriented tools.

Cytoscape - http://www.cytoscape.org/
• “An open source software platform for visualizing molecular interaction networks and biological pathways and integrating these networks with annotations, gene expression profiles and other state data.”
• Desktop application, but accesses web resources.
3. Network Formats

Import and Export of Network File Formats

For All Network Types

Each network has a “sourceFormat” attribute that records the format in which it was imported or otherwise created. It is maintained by the NDEx Server and currently cannot be changed by the user.

SIF and Extended Binary SIF Networks

The simple interaction format is convenient for building a graph from a list of interactions. It also makes it easy to combine different interaction sets into a larger network, or add new interactions to an existing data set.

1. If a tab ‘\t’ character is found in the first line of the file. The SIF is treated as tab delimited, otherwise it is parsed as a whitespace delimited file.

2. In NDEx, each line in a SIF network file is mapped to a NDEx edge object. The “relationship type” field in that line maps to the predicate of that edge. Each edge has one source node and one or more target nodes depend on the number of target nodes in that line.

3. Each node field in the SIF file is mapped to an NDEx node object. If the value of the “node field” is a URI or CURIE formatted string, the NDEx server will create a BaseTerm object based on the string and then create a Node to represent that base term. If the value of the “node field” is a simple literal text, no BaseTerm will be created, only a Node will be created and the “name” attribute of the node will have the value of the “node field”.

4. If the SIF file is an Extended Binary SIF file, a header line will define columns that are treated in the following manner:
   1. the “INTERACTION_PUBMED_ID” field will be used to create linked Citation objects.
   2. “PARTICIPANT_NAME” field will be used to populate the “name” attribute of the node.
   3. “UNIFICATION_XREF” field will be used to create an alias of a node.
   4. “RELATIONSHIP_XREF” field will be used to create related terms of a node.
   5. The “NAME” field in the Extended Binary SIF Property header will be used to set the name of the network. “ORGANISM” and “URI DATASOURCE” are treated as properties of the network.

OpenBEL Networks

The OpenBEL Language
OpenBEL ([www.openbel.org](http://www.openbel.org)) is the public standard for the BEL language. It is designed to represent scientific findings by capturing causal and correlative relationships in context, where context can include information about the biological and experimental system in which the relationships were observed, the supporting publications cited and the curation process used.

A BEL document is a set of statements representing specific assertions from cited information resources. Statements are, in most cases, triples with context annotations. The most common type of context annotation are specialized structures to cite specific supporting evidence from knowledge sources, but a more general mechanism allows the annotation of biological contexts such as species, cell type, or cell line. When encoded as a network, a BEL document may have multiple edges of the same type between two nodes, each edge representing a different assertion from a different citation.

BEL documents are not primarily intended as a format for biological inference, but rather as a means to store reusable facts in a form that is well suited to the *assembly* of purpose-built biological models. Assembly can be automated or may be the result of manual selection and incorporation of findings to produce a specialized model. The choice of assembly algorithm and parameters will lead to different output models for the same input BEL documents.

A particular form of assembled biological model suitable for some types of qualitative causal reasoning and for visualization is the “Knowledge Assembly Model” (KAM). NDEX networks are in principle capable of expressing KAM structures, but as of NDEX v1.2, there are no examples of KAMs in NDEX.

BEL is distinct from many other biological representation schemes in that it employs a system in which all concepts referenced in statements, such as protein abundances, complexes, modified proteins, or reactions are represented by functional composition of terms. This system is supported directly in NDEX networks using FunctionTerm network elements.

BEL documents are expressed in:

- XBEL, an XML format.
- BELScript, a line-oriented text format designed for human readability and composition.
- BEL RDF

NDEX currently supports import and export utilities for XBEL. The following section describes the rules used to transform BEL documents to and from NDEX Networks and XBEL.

**NDEX Import rules**

XBEL is an XML format in which XML nodes representing BEL statements are grouped by nested nodes that set the biological and citation context annotations for each statement that they contain. The context annotations from outer contexts apply to the statements of inner contexts unless specifically contradicted by annotations in inner contexts.

The following rules are applied based on the type of XML node processed:

- **Header**
  - name, description and version are mapped to Network.name, Network.description and Network.version respectively.
  - “copyright”, “contactInfo” and “Disclaimer” are stored as network properties.
  - Author list in AuthorGroup are flattened and each author name is stored as an individual property in the network. LicenseGroup is stored in the similar way.
• **NamespaceGroup**
  o Elements are stored as Namespace objects in NDEx network.

• **annotationDefinitionGroup**
  o internalAnnotationDefinition
    ▪ the “id” attribute is mapped to a Namespace object.
    ▪ “description” and “usage” are stored as properties in the Namespace object.
    ▪ “listAnnotation” elements are flattened and stored as properties in the Namespace object.
  o annotationDefinitionGroup
    ▪ Each element is stored as a Namespace object in the Network.

• **statementGroup**
  o If element “name” or “comment” exists in statement group.
    ▪ if a citation exists in the annotationGroup at the same level, “name” and “comment” are treated as properties of the citation.
    ▪ if a support exists in the annotationGroup at the same level, “name” and “comment” are treated as properties of the support.
    ▪ otherwise “comment” are stored as properties for each statement in the current statementGroup and it will be also passed on to the next level of statementGroup. “name” will be ignored in this case.

• **annotationGroup**
  o evidence is mapped to a Support object in the Network.
  o citation is mapped to a Citation object in the Network.
  o annotations are stored as NDExPropertyValuePair objects on each edge (or node if the statement is mapped to a orphan node).

• **statement**
  o Case 1: statement has subject, object, and predicate
    1. statement maps to NDEx Edge element
  o Case 2: statement does NOT have object
    1. statement maps to NDEx Node element
    2. node may be an “orphan” with no edges, or possibly other edges will reference the node.
  o Case 3: statement object is a statement expression, S2
    1. statement object is encoded by a node that represents a ReifiedEdgeTerm
    2. The ReifiedEdgeTerm references an edge that is created based on S2
A comment attribute of a statement is stored as a property of the Network element that it is mapped to, i.e. either an edge or node.

**XGMML Networks**

The XGMML standard is defined by the Cytoscape application. The version of XGMML exported by different versions of Cytoscape are annotated with version strings. The current version of Cytoscape produces an XML document in which the `<graph>` element has a property of `cy:documentVersion="3.0"`

**Handling of XGMML Network Properties**

Properties of a network in XGMML are stored in several places within the document. Some of these properties are shared by all XGMML files.

The main `<graph>` element has the following properties in XGMML 3.0:

- `id=<id of the graph at the time it was exported from Cytoscape>`
- `label=<label on the graph at the time it was exported from Cytoscape>`
- `directed="1"`  
  - whether the edges should be treated as directional
- `cy:documentVersion="3.0"`  
  - XGMML version

The `<graph>` element also has these constant properties, common to all XGMML 3.0 files. Note that the URI for the XGMML namespace does not respond as of the NDEx v1.2 release.

- `xmlns:dc="http://purl.org/dc/elements/1.1/"`  
  - Dublin Core namespace
- `xmlns:xlink="http://www.w3.org/1999/xlink"`
- `xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"`  
  - RDF namespace root
- `xmlns:cy="http://www.cytoscape.org"`  
  - Cytoscape namespace root
- `xmlns="http://www.cs.rpi.edu/XGMML"`  
  - XGMML namespace

Within the `<graph>` element is an attribute: `<att name="networkMetadata">`  
This contains RDF that expresses properties of the network using standard ontologies, especially Dublin Core. Properties typically include:

- `<dc:type>`
A type descriptor of the network – intended for semantic categories, such as “Protein-Protein Interaction”.

- `<dc:description>`
  - NOT SUPPORTED in Cytoscape, always outputs “N/A”
- `<dc:identifier>`
  - Some standard identifier for the network, defaults to “N/A”
- `<dc:date>`
  - Creation date of network
- `<dc:title>`
  - Title of network.
  - Often identical to label property of `<graph>`, but not clear that this is always true
- `<dc:source>`
  - Source of the network
  - All XGMML networks exported from Cytoscape list [http://www.cytoscape.org/](http://www.cytoscape.org/) as the source
- `<dc:format>`Cytoscape-XGMML`<dc:format>`
  - All XGMML networks have this value for their dc:format attribute

An XGMML network may have additional attribute `<att>` elements within the `<graph>` element. In XGMML 1.1, a number of graphics properties of the entire network are expressed as attributes, such as:

```xml
<att type="real" name="GRAPH_VIEW_ZOOM" value="0.41322728443244305"/>
```

In XGMML 3.0, a separate `<graphics>` element within the `<graph>` element separates the graphic `<att>` elements of the network from other attributes.

**Treatment of XGMML Properties**

**XGMML->NDEx**

- All Graphics attributes are ignored as of NDEx v1.2
  - Note that if a particular XGMML network has general properties that, by name, imply that they are graphics attributes, they are handled as any other property. For example, a node might have a general property ‘color = blue’ set by the user, but that is not encoded as a `graphics attribute`, as it would if it had been set in Cytoscape using graphic attribute facilities.
- XGMML RDF in the networkMetadata attribute are stored as NDEx Network properties
• (Note that the stored properties reference the Dublin Core namespace (dc) when used)
  • All other XGMML graph attributes are stored as NDEx Network properties

NDEx->XGMML
• No presentation properties are output to XGMML.
• All properties recognized as networkMetadata are expressed in the RDF section
• All other NDEx Network properties are expressed as attributes of the <graph> section

Special XGMML Properties Mapped to Attributes in NDEx Network Objects
• name
  o XGMML->NDEx
    ▪ if : dc:title exists in networkMetadata<att> then dc:title -> Network.name
    ▪ else if name attribute in networkMetadata<att> then name -> Network.name
    ▪ else if label property in <graph> node then label -> Network.name
    ▪ else: filename -> Network.name
  o NDEx ->XGMML
    ▪ Network.name ->dc:title

• description
  o dc:description<->Network.description

• version
  o dc:version<->Network.version

• UUID
  o XGMML->NDEx
    ▪ if NDEX:UUID exists as an <att>: **NDEx will ignore this attribute and assign a new UUID**
  o NDEx ->XGMML
    ▪ Network.UUID -><graph><att>NDEX:UUID

BioPAX Networks
NDEx uses BioPAXPAXtools to parse each imported BioPAX network into an org.biopax.paxtools.model.Model object, and then transforms the Paxtools Model object into an NDEx network.
Translation Rules:
• The xmlBase attribute of the Paxtools model is stored as a “xmlBase” property in the Network.

• Each BioPAXElement object is mapped to an NDEx node.
  o BioPAX type is stored in the “ndex:bioPAXType” property of the NDEx Node.
  o For each property on that BioPAXElement:
    ▪ If the value of the property is a BioPAXElement object, create an Edge.
      ▪ The subject Node of the Edge is based on this BioPAXElement node
      ▪ The predicate of the Edge is a BaseTerm derived from the name of the property.
      ▪ The object Node of the Edge is based on the value of the property
    ▪ If the value of the property is a literal value, create an NDExPropertyValuePair object and add it to the properties of the Node.

• For each Xref element, in addition to creating a Node, NDEx adds additional objects to ensure that the citations and controlled vocabulary references for the Network will be handled consistently with other Networks.
  o Each PublicationXref will result in a corresponding Citation object linked to the annotated Node.
  o Each UnificationXref will add a value to the aliases attribute for the annotated Node linking to a corresponding BaseTerm.
  o Each RelationshipXref will add a value to the relatedTerms attribute for the annotated Node, linking to a corresponding BaseTerm.
Exporting Networks

Overview

- NDEx networks can be exported as downloadable files.
- Because some export operations may take minutes to execute, exporting is handled as a background task, similar to the processing of upload network files.
- Network files can be exported in their native format (default option) or stripped down to the simpler SIF format: the convenience of exporting a file in SIF format is that it can be opened and read in Microsoft Excel.

Selecting a Network For Export

- In the network display page, click the Actions button and select Export Network as File
- Click Create Task to export your file in the default native format (XBEL in this example) or use the dropdown menu to export in SIF format.
Viewing the Export Status and Downloading Exported Networks

- The task is displayed on the right of the user’s account page, indicating its status as it progresses from queued to staged, to processing and then to completed.
- When the task is completed, a download link will appear.
- Click the link to download the exported file.

Uploading Networks

- In your MyAccount page, Click the blue Actions button, then select the Upload Networks option.
• Select the file to upload by using the Choose file button in the upper left corner.

• Click on the green Upload button to begin the process.
• Alternatively, select multiple files and upload them at once by clicking Upload all.
• Once the process is complete, a new task will be created. The task is a pending action that the NDEx system will resolve.
• You can see a “network loading” task at the bottom of the screen.

Note on Cytoscape XGMML

Depending on the version you are using, Cytoscape may default to exporting XGMML files with the extension .xml.

You must either specify the .xgmml extension at export time or rename the file before importing to NDEx. (Your operating system may warn you about changing the extension). The upload will fail if the extension is .xml.

XGMML presentation properties (layouts, styles, graphic properties) are not imported to NDEx.

• You can also view a task on the account page.
The status will change during the upload process. When the task is done, you will see the status change to completed. At this point, the network will be in the system and appear at the top of the networks list in your account page.

Saving a Query Result

- In this example, the signed-in user views a large network and finds a neighborhood based on query terms.
• The query result can be saved as a new network using the Save Current Subnetwork option in the Actions menu.

A dialog window will ask to confirm before saving.

• The new network is now visible on the user’s Account page.
Editing the Network Profile

- As an admin of the new network you just saved, the signed-in user can modify the network profile information using the Edit Network Profile item in the Actions menu.
- Besides changing the network name and adding a description, you can also control the visibility and accessibility of a network that you administer.
- By default, your new subnetwork will be **PRIVATE**: this means you are the only one who can see it and the network will not be displayed when any other user runs a search.
- You can change the visibility to **DISCOVERABLE** if you want the network to be visible after a search but still want to control who has access to it. In this case, another user will have to request access to the network and you can decide whether you want to grant access or not.
- If you decide to make your network **PUBLIC**, everyone will be able to access it, even anonymous users (users that are not logged in to an NDEx account.)
4. NDEx Basics

Creating and Using an NDEx Account

- On the NDEx landing page, select “Sign in” in the top right corner and then choose “Click here to sign up”

![NDEx Welcome](image)

- A dialog box will appear. An account name, email address, and password are required to sign up. The system does not allow duplicate account names or email addresses.

![NDEx Sign Up](image)

“My Account” Screen

- Once signed up, you will land on your “My Account” page. This page has 2 tabs that will show all the Networks and the Groups to which the account has direct access and will look like the one in the following screenshot.
- In the image below, no networks are visible because the account has just been created.
• For the rest of this tutorial, we will use example accounts that have access to several different networks and groups

[Networks and Groups]

Dr. Evil
street

Edit Profile

• Click the blue “Actions” menu button on the left of the user page and select “Edit Profile”

[Edit Profile dialog]

• Fill out the fields in the displayed dialog box (none of the fields are required, so any may be left blank).
• Click submit to make the changes.
• (Note that if you do not already have an image URL, you can use an image hosting site such as [http://imgur.com](http://imgur.com) to store an image and then reference it from NDEx.)

Change Password
• Navigate to the your account page.
• Click on the blue “Actions” menu button

![Account Groups Image]

**Account Groups**

• On the account page, all groups in which the user is member of can be seen by clicking “Groups”
• Filter by name or role in group

![Requests and Tasks Image]

**Requests and Tasks**

• Requests and Tasks relative to your account activity are displayed in the respective panels on the right side of your account page.
• Push notifications are currently not enabled, so click on the refresh icon (indicated by arrow in image below) to update the list: alternatively, you can use the browser refresh button or press CTRL+R.
• Requests sent and received are displayed in separate tabs. You will receive a request ONLY if you are the admin of a group or network another user wants to join or access, respectively.
• Click on a received request to respond; click on a request you have sent to delete it.

• The display interface for tasks is very similar with the exception that a task can be deleted by clicking the trash can icon.

Sign Out

• The sign out action is located on the right end side of the navigation bar.
Obtaining and Granting Access to a Network

Requesting Access to a Network

- When viewing this user’s account, one of their networks is displayed because it’s visibility status is discoverable.

- Click on the entry to display the network.
- Only the profile information for the network is displayed
- Click on the Actions button on the left of the screen and select Ask for Access.

- A dialog box is displayed, allowing you to select the level of access requested and to optionally add a message for the network’s admins.

- The pending request is displayed in your MyAccount page.
• Once your request has been accepted (or refused) you will see a confirmation message and the network now appears in the display of networks to which you have access.
• The accepted request can be deleted by the user – clicking on the request displays a deletion dialog.

Granting access to a network

If you are the admin of a network, you will be receiving requests for access (for example from Susan Calvin) and the pending requests are displayed on your MyAccount Page.
Click on the pending request to display the dialog to respond to the request. You may choose to add a message for the requesting user.

Managing Network Access

If an account is the admin of a network, it can manage the access other accounts have to that network.

- Navigate to the desired network page.
- Select the Manage Access option in the actions menu.
A new screen will display with a list of all the accounts with access to the network.

- You can change the type of access and remove access on this screen. Remember to save your changes.
- Groups may also have access to a network.
- Additionally, you can use the simple search tools at the bottom of the page to find groups and users you may want to grant immediate access. Users who already have access will have a faded add button displayed.
Quick start guide

The NDEx Public Server includes a large number of networks that are marked as “PUBLIC” and are therefore accessible without signing in to a user account. Public networks can be found, viewed, and queried anonymously using the search bar provided in the NDEx Public Server’s landing page.

Searching for networks

- To search for networks, type cell cycle into the search box and click the magnifying glass or press enter:

  ![Search for networks](image1)

- The network search results page is displayed below and lists several public networks:

  ![Search results](image2)
Now, click on a public network to view it: let’s choose the BEL Framework Small Corpus Document… In the network display page, information about the network is available on the left and includes the counts of nodes and edges for the entire network, version, date of creation and UUID. Information also includes the username of the network’s administrators.

Running a query

To run a query on this network, use the text box in the query controls:

- You can enter terms to query the network and specify a depth: 1-Step finds only the immediate neighbors of the nodes.
- For example, type akt1, select a depth of 1-step and click the “Run Query” button: the query will find a neighborhood around all nodes that reference the akt1 term. As shown in the image below, the query has retrieved a subnetwork, a small neighborhood consisting of “73 nodes” and “74 edges”. Additional useful information (such as Citations) about nodes and edges can be obtained by analyzing the table below the graphic representation.
Searching for users

- NDEx also allows to search for users or groups: for example, you can try to search for Users and type the term database: the result will be a list of users that have database in their name or profile description. Every database user has public networks that you can browse, explore and query.
Finding and Querying Networks

The public NDEx site includes a number of networks that are marked as “PUBLIC” and are therefore accessible without signing in to a user account. Public networks can be found, viewed, and queried but you cannot create new networks, upload, or download without signing in.

Some Public Networks on NDEx

• A selection of networks from NCI via Pathway Commons (SIF format)
• A selection of networks from Reactome via Pathway Commons (SIF format)
• OpenBEL (BEL format)
  o BEL small corpus
  o BEL large corpus

Searching for Networks Based on Title and Description

• Search text can be entered directly into the navigation bar.
• The current search mode is displayed to the left of the search box and defaults to “Networks”
• Type “cell cycle” into the search box and click the magnifying glass or press enter to search.
• The network search results page is displayed with several results, both public and discoverable.
• To view a discoverable network you will need to request access to its administrator.
• Click on a public network to view it: let’s choose the BEL Framework Small Corpus Document.

View a Network Found in a Search

• Information about the network is displayed on the left, including the counts of nodes and edges for the entire network, date of creation, UUID and links to the network administrators profiles.
• If a network is larger than 300 edges, no graphical display will be rendered and the main part of the page will display a table containing a sample of 500 randomly selected edges.
• The table has 3 tabs: one for Edges, one for Nodes and the last one for the Provenance history.

Advanced Search Capabilities

Advanced search capabilities (Lucene Indexing) allow users to search and query using keywords. Lucene Indexing allows:

• Network search by keywords
• The network search function in NDEx searches the “UUID”, “name” and “description” fields of a network.
• The network search function also searches the “base terms” and “node names” within a network.
• Network query by keywords
• The network query function in the network page searches “base terms” and “node names” in the network.

For more information and details about Lucene searches, please refer to the Lucene Documentation.

Query for a Neighborhood in a Network

Although when viewing a network the table only displays a sample of 500 Edges, any queries you perform will be always executed on the entire network. NDEx allows users to run 2 types of queries: Simple or Advanced.

Simple Query

In the text box in the query controls, you can enter terms to query in the network and the system will find a neighborhood around all nodes that reference those terms. The “depth” of the query defines the resulting network: 1-Step finds only the immediate neighbors of the nodes. For example:
• Type “akt1”, select a depth of “1-step” and click the “Run Query” button.

• The retrieved network is now a small neighborhood specified by the “akt1” term, consisting of “73 nodes” and “74 edges”. All the retrieved nodes and edges can be inspected in the table and additional filtering is possible using the text box at the top of each table’s column.

Advanced Query
As of NDEX v1.2, the Advanced Query feature allows users to query a network based on properties associated to its nodes and/or edges.

• Click the “Back to Original Network” button and then click Advanced Query on the right hand side.
As pictured below, the **Advanced Query** interface will show up, allowing you to search the network by filtering nodes and edges based on their properties. Properties can be identified by scrolling to the right in the Edges/Nodes table. You can select as many properties as you want on both Edges and Nodes. Properties and their values are case-insensitive.

- For example, type “Disease” in the property text box and “Breast Neoplasms” in the value text box, then click “Run Query”.
• The system will return a network consisting of nodes and edges where the Edges have the Disease property defined as Breast Neoplasm.

• In addition, the Advanced Query also allows you to query a network by predicate: click the “Back to Original Network” button, type “index: predicate” in the property text box, “positive_correlation” in the value text box and then click “Run Query”: 
This time the system will return a neighborhood consisting of 74 edges whose predicate value equals to “positive_correlation”.

Future developments will introduce auto-complete functions and drop down menus to easily view and select the properties of interest to be used in an advanced query.
5. Data Model

NDEx Network Data Model

Overview

NDEx Supports Diverse Representations of Biology

The NDEx network data model enables the storage of networks with diverse semantics, uploaded from files in a variety of source formats, including SIF, XGMML, XBEL, and BioPAX3. The intent is that the NDEx data model should partially integrate these diverse network formats to provide users and application developers with consistent handling of nodes, edges, namespaces and identifiers, citations, properties associated with nodes and edges, and network provenance history. The NDEx data model does not, however, standardize the representation of biology in the networks that it stores. The meaning of the relationships indicated by edges or the classes indicated by the types of nodes in a network may conform to a rich standard such as BioPAX3 or OpenBEL, or they may have ad-hoc meanings unique to the particular network. NDEx provides a common storage medium and access protocol, facilitating the use of diverse networks by applications but not limiting the semantics that they may express.

The intent in the design of the NDEx network data model and in any utilities for loading specific network formats is to fully preserve the information content of networks: a network file in a given format imported to NDEx should be equivalent (though not necessarily identical) to a network file output in a subsequent export operation using that format. As of NDEx v1.2, this intent is realized for SIF, BioPAX3, and OpenBEL: import-export 'round-trip' import-export cycles preserve the content but not the details of the structure of the original file. In contrast, the graphical and layout attributes for XGMML networks and for networks exchanged directly with Cytoscape via the CyNDEx app are not preserved. In these cases, these presentation aspects of the network are not stored in NDEx and are therefore not available when the network is subsequently retrieved or exported. This is an active area of development for NDEx in collaboration with the Cytoscape Community in which we are working to develop interchange standards that will enable applications, including NDEx, to flexibly handle diverse presentation schemes as modular aspects of networks. Details of the methods of encoding currently supported formats in the NDEx data model are described in a separate document, Handling of Network Formats, for SIF, XGMML, BioPAX3, and OpenBEL files.

NDEx is Extensible

Although NDEx provides both API and user interfaces to upload files in common formats (XGMML, XBEL, SIF, BioPAX3), the API also provides methods to create and query networks in a JSON format that is a serialization of the NDEx network data model. This enables researchers and developers to create and use networks with arbitrary semantics while still taking advantage of the common infrastructure supported by NDEx. For example, researchers might experiment with novel representations of RNA-RNA and RNA-DNA interactions using NDEx facilities for controlled vocabularies, citations, or terminology definition by functional composition. The resulting networks would benefit from NDEx-enabled applications for common functions such as basic visualization, indexing for search, or sharing and annotation. Specialized, modular applications (including ad hoc scripts) can then be constructed using the NDEx API to perform analyses and visualization that depend on the novel representation choices. This pattern of use
is intended to foster experimentation with representations with rapid, straightforward sharing and discussion of the representational strategies and analytic consequences.

**Goals of this Document**

In this document, the primary focus will be on the logical structure of the NDEx networks, but we will reference examples expressed in the JSON serialization that is used by the NDEx v1.2 API. We anticipate creating alternative serializations in the future, especially to enable incremental, streaming transactions involving networks, but all serializations will express the same logical structure.

One aspect of NDEx networks is described separately: the Network Provenance History structure associated with each network is explained in the [Network Provenance History document](#).

Aspects of the NDEx Network Data Model described in this document include:

- Reification of edges – enabling a node to represent an edge
- Functional term expressions – defining unique controlled vocabulary terms by functional composition of controlled vocabulary terms.
- Separation of presentation properties from other ‘user’ properties
- Detailed citations – augmented by ‘support’ objects that can specify particular text within a citation

**Network Accession in NDEx**

Networks stored on an NDEx server are assigned a 128-bit identifier that is a universally unique identifier (UUID). All networks are unique objects, regardless of what server they are created on.

The network UUID serves as an accession number across all NDEx servers. Networks can be accessed on their NDEx server by providing their UUID as a parameter to appropriate API methods.

The uniqueness of each network implies that copies of networks are different objects and will have a different UUID. It also facilitates the construction of network provenance histories that can be used to track the chain of sources and events leading to the current state of a given network.

**Objects of the NDEx Network Data Model**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetworkSummary</td>
<td>Brief summary information about the network</td>
</tr>
<tr>
<td>Network</td>
<td>The full network data structure</td>
</tr>
<tr>
<td>Namespace</td>
<td>Defines a reference to a controlled vocabulary, such as an external ontology</td>
</tr>
<tr>
<td>BaseTerm</td>
<td>Defines a controlled vocabulary term used in the network</td>
</tr>
</tbody>
</table>
FunctionTerm Defines a term used in the network by functional composition of other terms

ReifiedEdgeTerm Defines a term denoting an edge within the network for reference by other network elements

Node A vertex of the network graph structure

Edge An edge, relationship of the network graph structure

Citation A knowledge source – such as a journal article – that supports one or more Edges or Nodes.

Support Text supporting one or more Edge or Node objects, frequently also specifying the Citation that was the source of the text.

NdexPropertyValuePair Name-value pair with optional value datatype. An NdexPropertyValuePair includes deprecated attributes allowing it to optionally reference BaseTerm objects to specify controlled vocabulary terms. Unification with SimplePropertyValuePair is planned for future NDEx releases.

SimplePropertyValuePair Name-value pair.

NetworkSummary

NetworkSummary objects are a subset of Network objects. They are used to convey basic information about a network in API operations such as a simple GET of a network by id or when a list of NetworkSummary objects is returned as a search result.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>string</td>
<td>Text description of the network, same meaning as dc:description</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name or title of the network, not unique, same meaning as dc:title</td>
</tr>
<tr>
<td>creationTime</td>
<td>timeStamp</td>
<td>Time at which the network was created</td>
</tr>
<tr>
<td>modificationTime</td>
<td>timeStamp</td>
<td>Time at which the network was last modified</td>
</tr>
<tr>
<td>isComplete</td>
<td>boolean</td>
<td>Set to false while the network is being incrementally created or modified, true otherwise.</td>
</tr>
<tr>
<td>isLocked</td>
<td>boolean</td>
<td>Content modification permitted only if false</td>
</tr>
<tr>
<td>visibility</td>
<td>string</td>
<td>One of PUBLIC, PRIVATE, DISCOVERABLE. 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. DISCOVERABLE means that it can be found but that users without access must request permissions.</td>
</tr>
</tbody>
</table>
isPublished boolean (planned for NDEx v1.3) If true, network is permanently locked for content modification but access privileges can be altered.

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

presentationProperties list List of SimplePropertyValuePair objects: describes presentation properties of the network, such as stylesheet information controlling the display of classes of network elements.

### Network

A Network object contains all of the attributes of a NetworkSummary plus attributes that organize the eight types of NetworkElement objects that describe the content of the network. Each type of NetworkElement object is organized into a separate map (hash table, dict, dictionary, etc), indexed by element id.

Element ids are are unique within the network, but not globally unique. No two elements in a network will have the same element id, even if they are of different types. For example, all Edge objects are stored in a map that is the value of the edges attribute of the Network, indexed by unique element ids. The element ids present in any serialization or other encoding of the network outside of NDEx are not guaranteed to be preserved when the network is processed in any way or is stored in the NDEx Server. The NDEx Server and other applications are free to reassign element ids as they require, as long as the network remains internally self-consistent.

In the NDEx v1.2 implementation, the API requires that element ids are integer values.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespaces</td>
<td>map</td>
<td>Namespace objects by element id</td>
</tr>
<tr>
<td>baseTerms</td>
<td>map</td>
<td>BaseTerm objects by element id</td>
</tr>
<tr>
<td>functionTerms</td>
<td>map</td>
<td>FunctionTerm objects by element id</td>
</tr>
<tr>
<td>reifiedEdgeTerms</td>
<td>map</td>
<td>ReifiedEdgeTerm objects by element id</td>
</tr>
<tr>
<td>citations</td>
<td>map</td>
<td>Citation objects by element id</td>
</tr>
<tr>
<td>supports</td>
<td>map</td>
<td>Support objects by element id</td>
</tr>
<tr>
<td>nodes</td>
<td>map</td>
<td>Node objects by element id</td>
</tr>
</tbody>
</table>
Network Elements

Several of the classes of Network Elements (Namespace, Citation, Support, Node, Edge) include the attributes properties and presentationProperties, lists of property-value pair objects that enable the association of arbitrary, user or application-defined attributes with the Network Element. Presentation properties enable the separation of annotations specific to presentation and graphic display, such as layout coordinates or graphic styles.

Namespace

A Namespace object denotes a controlled vocabulary, such as an ontology, and must define either a prefix, a uri, or both.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Datatype Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>elementId Element id unique within the Network</td>
</tr>
<tr>
<td>properties</td>
<td>list List of NdexPropertyValuePair objects: describes optional (user-defined) attributes of the namespace.</td>
</tr>
<tr>
<td>presentationProperties list</td>
<td>List of SimplePropertyValuePair objects: optional attributes to describe graphic presentation of the namespace.</td>
</tr>
<tr>
<td>prefix</td>
<td>string The prefix string for abbreviated reference to this namespace.</td>
</tr>
<tr>
<td>uri</td>
<td>string The uri defining the namespace</td>
</tr>
</tbody>
</table>

Term

There are 3 types of Network Elements that are categorized as Term objects: BaseTerm, FunctionTerm, and ReifiedEdgeTerm. BaseTerm objects denote controlled vocabulary terms, terms in a defined Namespace. FunctionTerm objects denote concepts by functional composition of other Terms. ReifiedEdge objects enable reference to edges within the network by other network elements.

BaseTerm

BaseTerm objects denote terms in specified controlled vocabularies and may be used to define the meaning of Network Elements such as a Node or the relationship indicated by an Edge. BaseTerm objects specify the vocabulary by reference to a Namespace object. For example, a BaseTerm could denote a specific gene symbol in the HGNC official vocabulary of human genes.
FunctionTerm

FunctionTerm objects denote concepts by the functional composition of other Term objects. Both the function and the parameters are defined by reference to other terms, including other FunctionTerm objects. They provide a powerful mechanism for succinct representation of concepts that are inherently combinatorially explosive, such as modified protein species. The OpenBEL biological representation language defines all of its terms by functional composition and OpenBEL networks loaded in NDEx use FunctionTerm objects. FunctionTerm objects can also be employed in the development of novel representation schemes.

ReifiedEdgeTerm

ReifiedEdgeTerm objects denote Edge objects in the Network by reference to the element id of the Edge. This provides the mechanism by which a Node in anEdge can represent another Edge. For example, a ReifiedEdgeTerm object can be used to express a relationship between a concept and a relationship, such as where the abundance of a protein affects the causal relationship between two other entities, as in “A -| (B -> C)”. 

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>element id</td>
<td>Element id of this object within the Network</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>type is always ‘BaseTerm’</td>
</tr>
<tr>
<td>namespaceId</td>
<td>element id</td>
<td>Element id referencing a Namespace object. If not specified, the BaseTerm is implicitly in a default namespace for the network</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>The identifier for this controlled vocabulary term (required)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>element id</td>
<td>Element id within the Network</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>type is always ‘FunctionTerm’</td>
</tr>
<tr>
<td>functionTermId</td>
<td>element id</td>
<td>Element id of a BaseTerm object denoting the composing function of the FunctionTerm (required)</td>
</tr>
<tr>
<td>parameterIds</td>
<td>list</td>
<td>List of element ids where each element id refers to a Term object. The referenced Terms define the parameters of the functional expression. Unlike most lists of ids used in NDEx objects, the order of the parameter ids is meaningful – they are the ordered arguments of the function. Any algorithm manipulating FunctionTerm objects must preserve this order.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>element id</td>
<td>Element id of a BaseTerm object denoting the composing function of the FunctionTerm (required)</td>
</tr>
<tr>
<td>parameterIds</td>
<td>list</td>
<td>List of element ids where each element id refers to a Term object. The referenced Terms define the parameters of the functional expression. Unlike most lists of ids used in NDEx objects, the order of the parameter ids is meaningful – they are the ordered arguments of the function. Any algorithm manipulating FunctionTerm objects must preserve this order.</td>
</tr>
</tbody>
</table>
### Citation

A Citation object describes a knowledge source – such as a journal article – that supports one or more a Edge or a Node objects in the Network. Citation objects have a similar intent to a BioPAX “PublicationXref”. The Citation must at define either an identifier for the knowledge source or a title.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>element id</td>
<td>Element id within the Network.</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>type is always ‘Citation’</td>
</tr>
<tr>
<td>properties</td>
<td>list</td>
<td>List of NdexPropertyValuePair objects: describes optional (user-defined) attributes of the Citation.</td>
</tr>
<tr>
<td>presentationProperties</td>
<td>list</td>
<td>List of SimplePropertyValuePairobjects: optional attributes to describe graphic presentation of the Citation.</td>
</tr>
<tr>
<td>identifier</td>
<td>string</td>
<td>A string identifying the knowledge source cited.</td>
</tr>
<tr>
<td>idType</td>
<td>string</td>
<td>Indicates the type of the identifier, default is ‘URI’. The possible formats for the identifier are not constrained, but a definitive URI or DOI is best practice, when possible. Examples of alternative identifiers could include a journal citation string or an identifier from a particular database of experimental data or processed results.</td>
</tr>
<tr>
<td>title</td>
<td>string</td>
<td>The title of the knowledge source, same semantics as dc:title. (optional)</td>
</tr>
<tr>
<td>contributors</td>
<td>list</td>
<td>List of strings in which each string identifies an individual who is a contributor to the cited knowledge source. (optional)</td>
</tr>
</tbody>
</table>

### Support

A Support object contains text that supports one or more Edge or Node objects. Supports typically reference Citation objects to indicate the source of the text. OpenBEL is an example of a representation language which uses detailed supporting references, but the Support elements in the NDEx data model is available for use by any Network.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>element</td>
<td>Element id within the Network</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>type is always ‘Support’</td>
</tr>
<tr>
<td>text</td>
<td>string</td>
<td>free text describing the evidence for Edge or Node objects that reference the Support, typically taken as an extract from the abstract or full text of a Citation</td>
</tr>
<tr>
<td>citationId</td>
<td>element</td>
<td>Element id of a Citation object, indicating that the text is derived from the Citation</td>
</tr>
</tbody>
</table>

**Edge**

An Edge object encodes the relationship between two Node objects in the Network. An Edge can be annotated with Citations, Supports, and user-defined properties and presentationProperties.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>element</td>
<td>Element id within the Network</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>type is always ‘Edge’</td>
</tr>
<tr>
<td>properties</td>
<td>map</td>
<td>List of NdexPropertyValuePair objects: describes optional (user-defined) attributes of the Edge.</td>
</tr>
<tr>
<td>presentationProperties</td>
<td>map</td>
<td>List of SimplePropertyValuePair objects: optional attributes to describe graphic presentation of the Citation.</td>
</tr>
<tr>
<td>subjectId</td>
<td>element</td>
<td>Element id referencing a Node object. Defines the subject, or ‘source’ in the relationship denoted by the Edge</td>
</tr>
<tr>
<td>predicateId</td>
<td>element</td>
<td>Element id referencing a BaseTerm object. Defines the relationship denoted by the Edge</td>
</tr>
<tr>
<td>objectId</td>
<td>element</td>
<td>Element id referencing a Node object. Defines the object, or target in the relationship denoted by the Edge</td>
</tr>
<tr>
<td>citationIds</td>
<td>list</td>
<td>List of element ids of Citation objects supporting the Edge</td>
</tr>
<tr>
<td>supportIds</td>
<td>list</td>
<td>List of element ids of Support objects supporting the Edge</td>
</tr>
</tbody>
</table>

**Node**

A Node object encodes an entity, a concept – a thing that the Network is about. The meaning of the Node is defined by its attributes ‘representsId’, ‘aliasIds’, and ‘relatedTermIds’. The ‘name’
attribute of the node can specify a default label for display to users. A Node can also be annotated with Citations, Supports, and user-defined properties and presentationProperties.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>element</td>
<td>Element id within the Network id</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>type is always ‘Node’</td>
</tr>
<tr>
<td>properties</td>
<td>list</td>
<td>List of NdexPropertyValuePair objects: describes optional (user-defined) attributes of the Citation.</td>
</tr>
<tr>
<td>presentationProperties</td>
<td>list</td>
<td>List of SimplePropertyValuePair objects: optional attributes to describe graphic presentation of the Citation.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>A preferred display name for the Node. Some nodes may not have a representsId attribute, in which case the name may also be the closest approximation of the meaning of the Node – relying on ad hoc interpretation by humans or applications.</td>
</tr>
<tr>
<td>representsId</td>
<td>element</td>
<td>Element id of a Term defining the primary meaning of the Node</td>
</tr>
<tr>
<td>aliasIds</td>
<td>list</td>
<td>List of element ids of Term objects, each of which denotes an equivalent concept to the primary meaning of the Node. For example, a gene symbol and a gene id may be aliases, denoting exactly the same concept. A protein id, however, is not an alias for a gene symbol. This attribute has the same intention as a BioPAX aliasXREF.</td>
</tr>
<tr>
<td>relatedTermIds</td>
<td>list</td>
<td>List of element ids of Term objects, each of which denotes a related concept to the primary meaning of the Node. For example, a gene symbol and the protein id for its gene product would be related terms, denoting concepts that are related but not identical. This attribute has the same intention as a BioPAX relatedXREF.</td>
</tr>
<tr>
<td>citationIds</td>
<td>list</td>
<td>List of element ids of Citation objects supporting the Node</td>
</tr>
<tr>
<td>supportIds</td>
<td>list</td>
<td>List of element ids of Support objects supporting the Node</td>
</tr>
</tbody>
</table>

**NdexPropertyValuePair**

The NdexPropertyValuePair object encodes property-value pairs that can be simple string-string pairs with an optional the dataType attribute to specify the interpretation of the value string.

As of NDEx v1.2, this data structure also supports attributes to reference controlled vocabulary terms by element ids of BaseTerm objects for the predicate (property), the value, or both. These attributes are deprecated and will be phased out in subsequent releases.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
<td>type is always ‘NdxPropertyValuePair’</td>
</tr>
<tr>
<td>predicateString</td>
<td>string</td>
<td>The property in the property-value pair</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>The value in the property-value pair</td>
</tr>
<tr>
<td>dataType</td>
<td>string</td>
<td>Specifies the data type of the value, defaults to ‘String’ if this attribute is not set</td>
</tr>
<tr>
<td>predicateId</td>
<td>element id</td>
<td>(deprecated) The element id of a BaseTerm representing the property in the property-value pair.</td>
</tr>
<tr>
<td>valueId</td>
<td>element id</td>
<td>(deprecated) The element id of a Term representing the value in the property-value pair</td>
</tr>
</tbody>
</table>

**SimplePropertyValuePair**

The SimplePropertyValuePair object encodes property-value pairs that are simple string-string pairs. As of NDEx v1.2, this data structure is used as elements of the presentationProperties of Networks and network elements as described above.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
<td>type is always ‘SimplePropertyValuePair’</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>the name of the property</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>the property value</td>
</tr>
</tbody>
</table>
6. Metrics

**NDEx Content and Usage Metrics**

As of the v1.2 release, June 26, 2015, the NDEx public server (public.ndexbio.org) stored a total of 383 networks across 203 accounts.

The bulk of these networks were owned by 12 accounts corresponding to organizations that publish or aggregate network content.

**Network Formats**

<table>
<thead>
<tr>
<th>Format</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>XGMML</td>
<td>5</td>
</tr>
<tr>
<td>SIF</td>
<td>302</td>
</tr>
<tr>
<td>BEL</td>
<td>5</td>
</tr>
<tr>
<td>BIOPAX</td>
<td>31</td>
</tr>
</tbody>
</table>

**Network Sizes**

![Network Sizes Chart]
7. Provenance

**Network Provenance History**

The provenance history aspect of an NDEx network is used to document the workflow of events and information sources that produced the current network. API operations that create or update networks add default events to the provenance history. Applications can also explicitly modify the provenance history in order to customize events, controlling the granularity of events recorded and the level of detail captured.

**Motivation**

A network can represent assertions of biological relationships that are the results of experimental, analytic, or curation processes. Networks may in turn serve as inputs to further processes of analysis and model creation. If the workflow and dependencies on information sources are clearly documented, researchers may better understand the meaning of the relationships in the network and are better empowered if they wish to reproduce the analyses leading to the network. To achieve these goals, networks stored in NDEx can optionally include a provenance history aspect that can be accessed and managed via the NDEx API.

For example, a network might be derived by an algorithm which finds subnetworks based on experimental data mapped to entities in a reference network; in this case the application performing the analysis should record the analysis event in the provenance history of the output network, including references or descriptions of the algorithm used, the input experimental data, and a description of the input reference network.

For robustness, the provenance history stores descriptions of ‘ancestor’ networks and other information sources, not just links to those resources. This preserves the utility of the provenance history in situations in which some or all of the input information sources are unavailable or have been modified since they were used in the workflow. Researchers (or algorithms) can inspect the provenance history of the current network to address questions about the status of all of the inputs to the workflow.

**Related Work**

NDEx network provenance history is similar in intent to [Synapse Analytical Provenance](https://example.com).

**Provenance History Structure**

A provenance history is a tree structure containing ProvenanceEntity and ProvenanceEvent objects (Figure 1). 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.

**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

ProvenanceEvent
• endingAtTime
• timestamp
• has semantics of PROV:endingAtTime
• startingAtTime
• timestamp
• has semantics of PROV:endingAtTime
• inputs
• array of ProvenanceEntity objects
• has semantics of PROV:used
• properties
• array of SimplePropertyValuePair objects
Provenance History and Network Equivalence

The provenance history can be used to infer network equivalence, whether a given network stored in NDEx has the same content as another network or an external resource. This is valuable since in the general case, computing equivalence by algorithm may be computationally expensive or could require network format-specific knowledge.

Two networks on NDEx servers may be inferred to be equivalent if the following conditions are met:

- One is the ancestor of the other in their provenance histories.
- The events between the ancestor and descendent are all information preserving COPY operations.
- The ancestor has not been modified since the initial COPY operation.

Similarly, a network may be considered equivalent to an external source in either of the following cases:

- The network is the output of an UPLOAD event of a file derived from the external source and the external source has not been modified since the time of the upload.
- The network is an unmodified copy of a network meeting the above criteria.
Provenance Updates by NDEx API Operations

Seven REST API methods perform default updates to the provenance history of a network. They record basic information about the network (e.g. number of edges, nodes, title, description, version) and the event (e.g. type, time, username, first name, last name).

addNamespace /network/{networkId}/namespace
Records the added namespace name and value as an event property.

setNetworkProperties /network/{networkId}/properties
Records properties as event properties.

setNetworkPresentationProperties /network/{networkId}/presentationProperties
Records presentation properties as event properties.

updateNetworkProfile /network/{networkId}/summary
Records network name, description, and version as event properties and displays the current state as node properties.

createNetwork /network/asNetwork
No additional information recorded.

createNetwork /network/asPropertyGraph
No additional information recorded.

uploadNetwork /network/upload
Records filename as an event property.

Network Updates that do NOT Modify Provenance History

The provenance history is NOT updated when:

- Network membership information is changed or when network visibility (e.g. PUBLIC, PRIVATE, or DISCOVERABLE) is changed.
- The provenance history is explicitly updated by the API.

Reading and Setting Provenance History
An application can read and alter the provenance history without adding any additional event to
the provenance history. The API methods are:

- `getProvenance /network/{networkId}/provenance`
- `setProvenance /network/{networkId}/provenance`

**Provenance Events vs. Network Modification Times**

Any operation that modifies the network, including changes to visibility or provenance also
changes the last modification date of the network.

Changes to network membership – what users have access to the network – do not modify the
network itself and so do not change either the modification date or provenance history.

**Properties of ProvenanceEntity and ProvenanceEvent objects**

The standard fields in ProvenanceEntity and ProvenanceEvent objects correspond to
relationships defined in the PROV-O ontology. Other property-value pairs can annotate these
objects to provide more information about the entities and events. Any ad hoc pair of strings can
be added as a property-value pair, and the properties used may be idiosyncratic to the recorded
events and entities. However, the use of properties defined in the Dublin Core (DC) metadata
annotations and the Provenance, Authoring and Versioning ontology (PAV) are preferred when applicable.

It is important to note a difference in the use of these ontologies in an NDEx provenance
structure and the original intent. A ProvenanceEntity is a description of the referenced object,
not the object itself. Therefore, a property such as “dc:title” that is asserted for a
ProvenanceEntity refers to the original entity that the ProvenanceEntity represents. The
provenance history references ancestor networks and other data sources but can also include
self-contained descriptions of those objects that capture their state at the time they were used.

**Dublin Core (DC) Properties**

- `dc:title`
- `dc:description`
- `dc:rights`
- `dc:rightsHolder`
- `dc:format`

**PAV Properties**

- `pav:retrievedFrom`
  - Direct retrieval – a COPY of the source network with no transformation of the content.
• **pav:importedFrom**
  * Import with some transformation, as in a file UPLOAD where the source data is processed to create the network.
  * The content reflects the external source but potentially has differences dependent on the import method.

• **pav:derivedFrom**
  * The network was generated by an operation that transforms the content of the source.

• **pav:sourceAccessedAt**
  * The network was generated by a transformation operation that consulted the source as part of the transformation.

• **pav:version**
  * The version of the current network.

• **pav:previousVersion**
  * The previous version. Note that this might be the version of a network that is not in the provenance history – a version could be created from new sources, not necessarily as a transformation of an earlier version.

**Provenance History Use Cases**

**Copying Networks**

In a copy operation, an application / utility creates a new network (the target) that encodes the same content as an existing network (the source).

In the resulting target provenance history, the root ProvenanceEntity represents the target and the copy operation is represented as a ProvenanceEvent of type COPY in which the output is the root entity and the input is a ProvenanceEntity representing the source.

The ProvenanceEntity representing the source and all of its prior entities and events are copied from the provenance history of the source.

Information stored in the provenance history about the source is intended to reflect the state of the source at the time of the copy and should not be updated to reflect subsequent changes in the source. Information about the source stored in the provenance history is thereby preserved, regardless of whether the source is later modified or deleted.

**Upload / Import Network File**
Upload is a special type of import, where the ProvenanceEntity for the source should store information about the uploaded file in the properties, such as the filename, file type, or data size.

**Network Query / Filter**

A network created by a query or other operation that retrieves part of the source is a common type of transformation operation. The new network is derived from the source.

**Editing Operations**

In any case where the source network has the same UUID as the target, the ProvenanceEvent is an edit of some type. Because the event can have both startingAt and endingAt properties, the editing process can span an arbitrary amount of time. The application managing the editing process can therefore control the granularity of the provenance history. For example, an editing application could represent a long sequence of edits in a verbose chain of events and intermediate states or it could simply keep updating the endingAt time as the edits continued. In both cases, the resulting provenance history would be a valid representation of the workflow, although one would capture greater detail than the other.

**Translation of Network Identifiers**

In the case where a utility creates a network that has content equivalent or homologous to the source but described in a different identifier system (such as gene ids replaced with corresponding gene symbols), an additional resource describing the identifier mapping is typically involved. In this case, the mapping resource is also an input to the ProvenanceEvent, and it is appropriate to use the property pav:sourceAccessedAt to describe the relationship.

**Merging Networks**

“Merging” in this context means a modification operation in which the information in network A is augmented by information coming from network B, or where a new network is created from both A and B. This creates a branched provenance history in which the ProvenanceEvent for the merge has two inputs, both network A and network B.
8. REST API

Working with Networks Using the NDEx Server API

Scope of this Document
This document covers the primary NDEx Server REST API operations needed to create scripts and applications working with networks in NDEx. All network-related operations on an NDEx Server may be performed via the REST API, but we recommend that most users use the NDEx Web User Interface to perform the following operations:

- Share a network with another user or a group
- Request access to a network
- Upload a network from a file in a supported format
- Export a network to a file in a specified format

Advanced developers may want to incorporate these operations directly into their applications, but in most cases it will be easier to take advantage of the existing user interface.

This document also intentionally omits discussion of the API methods that use the ‘PropertyGraphNetwork’ format. As of v1.2, that format is used by the CyNDEx Cytoscape App, but may be deprecated in v1.3, with key aspects of its functionality being shifted to client libraries in conjunction with new API methods using the upcoming CX (Cytoscape Cyberinfrastructure Network Exchange) format.

This document mentions Network and NetworkSummary objects that are extensively described in our NDEx Network Data Model document.

HTTP Transactions and Authentication

The NDEx Server API uses the four common HTTP transactions types: GET, PUT, POST, and DELETE.

API methods that create, modify, or delete networks always require authentication – a user must be authenticated based on credentials presented as part of the HTTP request. Further, a transaction will only succeed if the authenticated user has EDIT or ADMIN permissions for the specified network. In this document, these operations are all marked Requires Authentication.

API methods that query or otherwise access networks can be used without authentication, but in that case, they will only succeed for networks that are marked ‘PUBLIC’. If the same method is performed with an authenticated user, the operation will also succeed for networks for which the user has READ, EDIT, or ADMIN permissions.
Client Libraries

Client libraries facilitate the creation of NDEx-enabled applications. They manage the handling of authentication and other aspects of HTTP transactions and provide convenience methods to invoke NDEx Server API methods.

NDEx Java Client

The NDEx Java Client is available from the GitHub repository https://github.com/ndexbio/ndex-java-client.

The class NdexRestClient provides basic objects and methods to manage authentication and to perform GET, POST, PUT, and DELETE operations to an NDEx Server.

The class NdexRestClientModelAccessLayer provides convenience methods corresponding to each of the NDEx Server API methods.

In this document, we will present the corresponding NdexRestClientModelAccessLayer method for each NDEx Server API method (where available).

Find a Network by Accession

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 the getNetworkSummary method.

getNetworkSummary

GET : /network/{networkId}

Retrieves a NetworkSummary object based on 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

An application can retrieve a list of NetworkSummary objects corresponding to networks matching a text query using the searchNetwork method. Networks are matched based on the text in the name and description fields, plus the strings of node names and controlled vocabulary terms used in the network. As of NDEx v1.2, the underlying text indexing and search is performed by a Lucene engine. The search can also be constrained to networks owned by a specified account. The NetworkSummary objects returned by the query provide useful information including the network UUID, name, description, and counts of nodes and edges.

searchNetwork

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.

The query can specify the following parameters:

searchString Required. A whitespace-delimited string of search terms that is handled according to Lucene search string protocol.

permission Optional. String set to either ‘ADMIN’, ‘WRITE’ or ‘READ’. If set to ‘WRITE’, the search will only return networks for which the authenticated user has permission to edit. By default, the search will return networks that are readable or which have been marked DISCOVERABLE.

includeGroups Optional. Boolean value, defaults to false. If a user is a member of a group and the group has permissions to a network, then the user can access the network according to those permissions. If includeGroups is true, the search will also return networks based on the authenticated user’s group memberships.

accountName Optional. String value. If the accountName parameter is provided, then the search will be constrained to networks owned by that account.

canRead Optional. Boolean value, defaults to false. By default, the search will return networks that are marked DISCOVERABLE as well as the networks that the user can read. But if the canRead parameter is true, the DISCOVERABLE networks will be excluded.

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)

searchNetworkByPropertyFilter
POST : /network/searchByProperties

Requires Authentication

This method returns a list of NetworkSummary objects in no particular order which have properties (metadata) that satisfy the constraints specified by a posted JSON query object. The query object has the following format:

```
{
    "properties":
    [
    {"propertyName": string,
    "value": string},
    ... ]
    "admin": string,
    "limit": integer
}
```
properties Required. A list of objects associating a propertyName with a value. Networks that have at least one matching property-value pair in their properties will be returned by the search. As of NDEx v1.2, the matching of property values is limited to exact string equivalence.

admin Optional. String value. When admin has a null value, the method returns all networks that meet the search criteria. If an admin value is provided, it is treated as an account name and only networks owned by that user account (and meeting search criteria) will be returned.

limit Optional. Integer value. If “limit” has a value n where n>0, up to n networks will be returned in the result. n<=0 means no size limit in the result.

Get a Network

The getCompleteNetwork method enables an application to obtain an entire network as a JSON structure. 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 before attempting to use getCompleteNetwork. As an optimization, networks that are designated read-only (see Make a Network Read-Only below) are cached by NDEx for rapid access.

getCompleteNetwork

GET : /network/{networkId}/asNetwork

This method retrieves the entire network specified by ‘networkId’ as a Network object, including the information in the NetworkSummary for the network.

Java Client Method:

public Network getNetwork(String id)

Query a Network

queryNetwork

POST : /network/{networkId}/asNetwork/query
Retrieves 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 a POSTed JSON query object with the following attributes:

**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.

**depth** Integer value between 1 and 3. Sets the maximum number of traversal steps from
the initial nodes.

The subnetwork is returned as a JSON Network object containing the selected edges plus any
other network elements relevant to the edges.

**Java Client Methods:**

```java
public Network getNeighborhood(
    String networkId,
    String searchString,
    int depth)
```

```java
public Network getNeighborhood(
    String networkId,
    SimplePathQuery query)
```

**queryNetworkByEdgeFilter**

**POST**: `/network/{networkId}/asNetwork/prototypeNetworkQuery`

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
eedgeFilter 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**” : 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:

- **“ndx:nodeName”** : The value in the propertySpecification is matched vs. the name of the node.
- **“ndx: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.
- **“ndx: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” : <string>,
  “value” : <string>
}
```

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

Query JSON:

```
{  
  “nodeFilter” : {   
    “propertySpecifications” : [ <PropertySpecifications>],   
    “mode” : <mode>  
  },   
  “edgeFilter” : {   
    “propertySpecifications” : [ <PropertySpecifications>]  

```
getEdges
GET : /network/{networkId}/edge/asNetwork/{skipBlocks}/{blockSize}

This method retrieves a subnetwork of the network specified by ‘networkId’ based on a ‘block’ of edges, where a ‘block’ is simply a set that is contiguous in the network as stored in the specific NDEx Server. The maximum number of edges to retrieve in the query is set by ‘blockSize’ (which may be any number chosen by the user) while ‘skipBlocks’ specifies the number of blocks of edges in sequence to ignore before selecting the block to return. The subnetwork is returned as a Network object containing the edges specified by the query plus all of the other network elements relevant to the edges.

This method is used by the NDEx Web UI to sample a network, enabling the user to view some of the content of a large network without attempting to retrieve and load the full network. It can also be used to obtain a network in ‘chunks’, but it is anticipated that this use will be superseded by upcoming API methods that will enable streaming transfers of network content.

Java Client Method:
public Network getEdges(String id, int skipBlocks, int edgesPerBlock)

Get the Provenance History for a Network

getProvenance
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 for a detailed description of this structure and best practices for its use.

Java Client Method:
public ProvenanceEntity getNetworkProvenance(String networkId)
Create a Network

createNetwork
POST : /network/asNetwork
Requires Authentication

This method creates a new network on the NDEx Server based on a POSTed Network object. An error is returned if the Network object is not provided or if the POSTed Network does not specify a name attribute. An error is also returned if the Network object is larger than a maximum size for network creation set in the NDEx server configuration. A NetworkSummary object for the new network is returned so that the caller can obtain the UUID assigned to the network.

Java Client Method:

```
public NetworkSummary createNetwork(Network network)
```

Update the Network Profile Information

updateNetworkProfile
POST : /network/{networkId}/summary

Requires Authentication

This method 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’, ‘version’, and ‘visibility’.

Java Client Method:

Note the Java convenience method takes as an argument a NetworkSummary object populated with only the profile attributes that should be updated:

```
public NetworkSummary updateNetworkSummary(
    NetworkSummary networkSummary,
    String networkId)
```
Update an Entire Network

updateNetwork
PUT : /network/asNetwork
Requires Authentication

This method updates an existing network with new content. The method takes a Network JSON object as the PUT data. The Network object must have its UUID property set in order to identify the network on the server to be updated. This condition would already be satisfied in the case of a Network object retrieved from NDEx. This method errors if the Network object is not provided or if its UUID does not correspond to an existing network on the NDEx Server. It also errors if the Network object is larger than a maximum size for network creation set in the NDEx server configuration. A NetworkSummary JSON object corresponding to the updated network is returned.

Java Client Method:
public NetworkSummary updateNetwork(Network network)

Update the Properties of a Network

setNetworkProperties
PUT : /network/{networkId}/properties

Requires Authentication

Updates the ‘properties’ field of the network specified by ‘networkId’ to be the list of NdexPropertyValuePair objects in the PUT data.

Modify the Provenance History for a Network

setProvenance
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:

```java
public ProvenanceEntity setNetworkProvenance(
    String networkId,
    ProvenanceEntity provenance)
```

---

**Make a Network Read-Only**

The ‘readOnly’ status of a network can be controlled by an associated system flag that can be set via a general purpose API method. When the readOnly status of a network is true, it cannot be modified by any API methods. (Note that changing the permissions of a network – such as sharing it with another user – does not constitute a modification of the network).

An additional effect of making a network readOnly is that it enables the NDEx Server to optimize the storage and indexing of the network. In NDEx v1.2, when a network is flagged as readOnly, the NDEx Server initiates a background task to cache a JSON serialized version of the network for rapid retrieval, thereby making retrieval of the entire network dramatically faster. (But not changing the behavior of queries that retrieve subnetworks). The cached file is removed if the readOnly flag is reset to false.

**setNetworkFlag**

GET : /network/{networkId}/setFlag/{parameter}={value}

Requires Authentication

Set the system flag specified by ‘parameter’ to ‘value’ for the network with id ‘networkId’. As of NDEx v1.2, the only supported parameter is `readOnly={true|false}`

Java Client Method:

```java
public String setNetworkFlag(
    String networkId,
    String parameter,
    String value)
```

---

**Delete a Network**

**deleteNetwork**
DELETE : /network/{networkId}

Requires Authentication

Deletes the network specified by networkId. There is no method to undo a deletion, so care should be exercised. A user can only delete networks that they own.

Java Client Method:
public void deleteNetwork(String id)
9. CyNDEEx

CyNDEEx – The NDEx Cytoscape App – Prototype Tutorial

Overview

The NDEx Cytoscape App (CyNDEEx) provides a mechanism for you to take any network in Cytoscape and upload it to NDEx. You can also download networks from NDEx to Cytoscape. The goal of CyNDEEx is to be both easy to install and easy to use. As of June 16, 2015, the CyNDEEx was available for testing upon request, but please consult the online documentation at www.ndexbio.org for the latest status of CyNDEEx.

Why Use This App?

CyNDEEx allows you to transfer your networks between Cytoscape, the leading desktop solution for visualizing biological networks, and NDEx.

What Sort of Networks Can I Store?

Although NDEx was created with the intention of storing, sharing, and using biological networks, any kind of network can be stored in NDEx.

Getting Started

Before you can CyNDEEx, you must have Cytoscape 3.1.1 or better installed along with 64-bit Java 7. (As of June 2015, Cytoscape is not guaranteed to work with Java 8) You also must ensure that your JAVA_HOME environmental variable is set to Java 7.

1. To install Java 7, go here.
2. For instruction on how to set JAVA_HOME, go here.
3. To install Cytoscape 3.1.1 or later, go here.

Installing CyNDEEx

Now that you have Cytoscape installed and Java 7 setup correctly, it is time to install CyNDEEx. The latest version is currently alpha16 and it is available upon request. To request the App, please submit a CyNDEEx Request form. Once you have obtained the .jar install file from us, follow these steps:

1. Open up Cytoscape and select App Manager from the Apps menu...

2. Choose Install from File…
3. Select the .jar file that you obtained from us and install it.

4. Once the App Manager will indicate that the status of CyNDEx is installed, you can close the App Manager.

5. Now, under the Apps menu in Cytoscape, you will see an NDEx menu with 3 options.

Using CyNDEx

By default, you can anonymously download any “public” networks from the PUBLIC NDEx SERVER to Cytoscape using the CyNDEx you just installed; however, you will not be able to download “private” networks or upload any networks from Cytoscape to NDEx unless you are logged in to your NDEx Account. Therefore, in order to use the full functionalities of CyNDEx, you need an NDEx account. This tutorial was designed assuming that you already have an NDEx Account.

For instructions on how to create and NDEx Account, go here. Once you have registered your NDEx Account, proceed with the following instructions:

- Log in to NDEx

1. In Cytoscape, click the Apps menu, select NDEx and then click Change NDEx Source…

2. Make sure that NDEx Public is selected and click Edit
3. Enter your username and password and click Save.

4. Finally, click the Connect button in the bottom right corner and you will get a confirmation message that indicates that your username and password was correctly recognized.

- Download (import) networks from NDEx

1. Choose the Import Networks from NDEx menu item.

2. On the Find Networks screen, you can either select one of the networks on the list or search for more specific networks. For example, you can type the word metabolism in the
search box. Then make sure the “Metabolism of RNA” network is selected and press the Select Network button.

3. By default 25 edges out of 4344 edges are displayed. At this point, you could download the entire network, download these 25 edges, or do a query on the network to retrieve a subset of edges. Right now, we are going to download the entire network; to do so, make sure that the entire network radio button is selected and click Load.

4. After you begin your download, you will end up again on the Find Networks dialog. Select the “Metabolism of RNA” network and perform a query by entering in the word “PHAX” in the Query box and hitting enter (or clicking Run Query). In the bottom panel, you will see five edges. Now, change the name of the network to load to “PHAX in Metabolism of RNA”, confirm that the “Selected Subnetwork” radio button is selected, and click “Load”.
5. After this, you will once again find yourself in the Find Networks dialog. We are done loading networks and now want to view them in Cytoscape. Click the Done Loading Networks button in the bottom right corner so that we can see what we have imported in Cytoscape.

- **Visualize downloaded networks in Cytoscape**

Rearranging the network view windows in Cytoscape, we see that both networks we just downloaded look like they consist of 1 single node, although we know that one of them has over 4000 thousand edges and the other has 5 edges. The reason for this odd behavior is that neither of these two networks had any presentation properties set. Therefore, they appear in a simplified manner in Cytoscape.

In order to properly visualize a downloaded network, we need to apply a visualization layout.

1. To do so, just select a network, open the Cytoscape Layout menu and choose the Prefuse Force Directed Layout item.
2. After the layout has been applied, you will see much more elaborate looking networks…

Upload (export) networks to NDEx

Now, we are going to perform the final step in our tutorial: exporting a network to NDEx. Although we illustrate how to upload the “PHAX in Metabolism of RNA” network, the same exact procedure can be applied to any other networks you have been working on in Cytoscape, regardless of their provenance. Let’s get started!

1. First, select the “PHAX in Metabolism of RNA” network by clicking on its title bar, then open the Apps menu, select NDEx and click on Export Current Network to NDEx.

2. A window will appear which tells you which NDEx SERVER you are currently connected to, what users account that network will be exported to as well as the size of the network. The only option available is to change the name of the network (red arrow). In this case, we will keep the name of the network the same and click the Upload Network to NDEx button.
3. **Done!** You can now log in to your NDEx account and see that your “PHAX in Metabolism of RNA” network is displayed in your personal account page.

With NDEx and CyNDEx, you can easily share your networks with collaborators as well as use and improve networks shared by others. Tutorials and detailed user documentation for NDEx is available in the [Documentation](#) page on our website.
10. NDEx Sync

NDEx Sync: A Network Copier Utility

NDEx Sync is a command line utility that enables users to copy networks from one NDEx account (the source NDEx) to another (the target NDEx). Please consult the online documentation at [www.ndexbio.org](http://www.ndexbio.org) for the latest instructions for obtaining and using NDEx Sync.

Requirements

- Platform: Linux or MacOS
- Java 7 installed
- Network access to both the source NDEx and target NDEx.

License and Source Code

NDEx Sync is open-source software available under a BSD license. The source code is hosted on GitHub at https://github.com/ndexbio/ndex-sync

Running the NDEx Sync

NDEx Sync is used via the shell script /opt/ndex/lib/ndex-copier.sh

The script takes a single argument: a directory containing ‘copy plan’ files.

```
bash
  ndex-copier.sh /users/user12/my-copy-scripts
```

When run, the script reads and attempts to execute each copy plan file in the directory.

The NDEx Sync script can be run manually or can be executed periodically via cron or other scheduling facilities to copy new or modified networks from the source NDEx, creating or updating networks on the target NDEx.

How NDEx Sync Works

NDEx Sync is like a file-mirroring utility, but with an important difference: the copied networks are not exact duplicates of the source networks.

- Copied networks are assigned new UUIDs: every network stored in an NDEx server has a globally unique identifier and can be referenced by that identifier at its host NDEx.
- NDEx Sync updates (or creates, if necessary) the network’s provenance history, adding a “provenance event” that documents the fact of the copying.

The copied networks are therefore documented as distinct entities, copied at a specific time from a uniquely identified source. The provenance history provides a structure to document the events leading to the current state of a network. Applications using NDEx are not required to maintain the provenance history for networks that they manipulate, but it is encouraged as a standard practice and will be supported by NDEx utilities.

For each source network that is selected as a candidate for copying, NDEx Sync examines the provenance history of each network in the target account to determine:

- Was this target network copied from the source network?
- Is the target Out-Of-Date?
The default behavior of NDEx Sync is that it will copy the source network to the target account if there is no copy of the source network in the target account OR if the only copies are Out-Of-Date or have been modified.

**Update of Networks by NDEx Sync**

The default behavior of NDEx Sync is conservative, never overwriting or deleting any network in the target directory. This behavior can be overridden by the copy plan parameter `updateTargetNetwork`, specifying that NDEx Sync should `updatetarget` networks that are identified as unmodified, out-of-date copies of the specified source networks.

In an *update*, the target network keeps its UUID but its contents are replaced by the contents of the source network and the provenance history is handled in the same manner as in a default, non-update copy event. The updated network may be accessed by that UUID and any new request will obtain the updated content.

Using NDEx Sync to update networks is only appropriate for situations in which the target network is intended as a cache of the source, where users want to obtain the latest version of the source content and where they do not expect the content of the network to be consistent over time.

**Updates of Read Only Networks**

By default, updates will NOT be performed if the target network has `readOnly == true`. The `updateReadOnlyNetwork` configuration parameter in a copy plan overrides this behavior. This handles the case in which NDEx Sync is used to maintain a local copy of a remote resource and where the local copy is intended as a read-only reference.

**Out-Of-Date Criteria**

The criteria for “out-of-date” are as follows:

- Calculate latestSourceDate as the later of modification date and the last provenance history event end date for the source network.
- Calculate earliestTargetDate as the earlier of modification date and the last provenance history event end date for the target network.
- if latestSourceDate > earliestTargetDate, target is out-of-date

**Last Modification Date**

- The lastModificationDate field of a network is updated when:
  - There is a change to any network element, including properties, presentation properties
  - There is a change to intrinsic special “profile” properties
    - name
    - description
  - The lastModificationDate does not update on:
    - Changes to provenance history
    - Changes to permissions
    - Change to read-only status
    - Change to visibility
What is Copied with a Network

- **Copied:**
  - All network elements, including properties, presentation properties are copied.

- **Not Copied:**
  - Permissions
  - Visibility
  - UUID
  - Modification time, creation time
  - readOnly status

- **Copied but modified:**
  - Provenance History

Copy Plans

NDEx Sync ‘copy plans’ specify:

- An account and credentials for the source NDEx.
- An account and credentials for the target NDEx.
- The criteria to select networks on the source NDEx, which can be one of:
  - A query to find networks matching search text.
  - A query to find networks administered by an account AND matching search text.
  - A list of network UUIDs.
- The **updateTargetNetwork** parameter
  - The possible values of **updateTargetNetwork** argument are “true” or “false”.
  - The default value of this argument (i.e., if is missing from the copy plan) is “false”.
  - If **updateTargetNetwork** is set to “true”, NDEx Sync should check whether the target server account specified in the copy plan contains a network that was copied earlier from the source server, and decide whether to update the network in the target server account or not. In case the network only exists in the source server account and not in the target account, the network gets copied to the target account.

- The **updateReadOnlyNetwork** parameter
  - The value of **updateReadOnlyNetwork** argument is “true” or “false”.
  - Default value (if the argument is missing from the copy plan) is “false”.
  - If **updateReadOnlyNetwork** is true and the target account specified in the copy plan has the Administrator privileges for the target network to be updated then the target network can be updated even if it is set to readOnly = true. In this case, NDEx Sync changes the read-only flag to false, updates the network, and changes the read-only flag back to true.
  - The **updateReadOnlyNetwork** parameter is only used if **updateTargetNetwork** is set to true.
**Notes on Updates:** NDEx Sync can only update networks in the target server account if the account specified by the username in the target element in the copy plan has Administration privileges for the networks to be updated.

**Query Copy Plan**
- Source networks are identified based on their title, description, or content matching a query string.
- The user account for the source must have read access to each source network.

In the example copy plan below, networks matching "cal*" are copied from the public NDEx to the user2 account on an NDEx running on the local machine.

- **queryString:** search text to find networks.
- **queryLimit:** a maximum number of networks to copy is specified.
  - This is useful largely as a brake on runaway copying – if the queryString matched some unanticipated, enormous number of networks, the script would still be limited.

```
{
    "planType": "QueryCopyPlan",
    "source": {
        "route": "http://www.ndexbio.org/rest",
        "username": "user1",
        "password": "pwd00123"
    },
    "target": {
        "route": "http://localhost:8080/ndexbio-rest",
        "username": "user2",
        "password": "pwd980098"
    },
    "queryString": "cal*",
    "queryLimit": "10",
    "updateTargetNetwork": "false",
    "updateReadOnlyNetwork": "false"
}
```

**Query Copy Plan with Account**
- **sourceAccount:** Source networks are limited to those administered by the specified account name.
- To copy all the networks for a given account, the **queryString** can be ""
In the copy plan example below, all networks (up to 10) from the user3 account are copied from the public NDEx to the user2 account on an NDEx running on the local machine.

```
{
    "planType": "QueryCopyPlan",
    "source": {
        "route": "http://www.ndexbio.org/rest",
        "username": "user1",
        "password": "pwd00123"
    },
    "target": {
        "route": "http://localhost:8080/ndexbio-rest",
        "username": "user2",
        "password": "pwd980098"
    },
    "queryString": "",
    "queryLimit": "10",
    "queryAccountName": "user3",
    "updateTargetNetwork": "false",
    "updateReadOnlyNetwork": "false"
}
```

Network ID Copy Plan

- **idList**: list of UUIDs to identify source networks.
- The user account for the source must have read access to each source network.

In this example, the network 5bca3218-28ca-11e4-9032-90b11c72ae9a is copied from the public NDEx to the user2 account on an NDEx running on the local machine.

```
{
    "planType": "IdCopyPlan",
    "source": {
        "route": "http://www.ndexbio.org/rest",
        "username": "user1",
        "password": "pwd00123"
    },
    "target": {
        "route": "http://localhost:8080/ndexbio-rest",
        "username": "user2",
        "password": "pwd980098"
    },
    "idList": [
        "5bca3218-28ca-11e4-9032-90b11c72ae9a"
    ]
}
```
Limitations

NDEx Sync can fail to copy very large networks. This is because it operates by first obtaining the all the data from the source network in a single REST request and then stores the data to the target NDEx in a single REST request. This simple strategy results in the in-memory instantiation of the entire network at the source NDEx, on the copying machine, and at the destination NDEx. Failure can occur when the process exceeds the available memory at any of these stages. Future versions of NDEx Sync will transition to use CX-based streaming, incremental NDEx server network transfer mechanisms as they become available, thereby limiting the memory footprint at each stage.