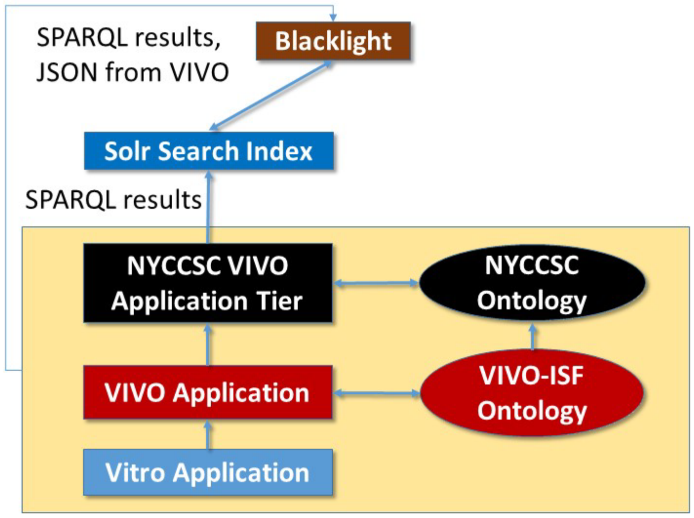
New York Climate Change Science Clearinghouse (NYCCSC) Technical Documentation

Overview and Architecture

This document provides an overview of the system components of the NYCCSC website, describes the open source technologies used in the NYCCSC project and provides installation steps.

The NYCCSC project relies on the VIVO[1] application, VIVO's built-in Solr[2] search index, and a Blacklight[3] front-end. The architecture overview diagram below describes how these different components fit together.



- Blacklight = Discovery front-end
    - Rails Gem
- Solr search index
    - VIVO's built-in search index, modified
- NYCCSC VIVO
    - Ontology representing climate change
    - Uses and extends VIVO ontology
    - Populates Solr search index using SPARQL queries

Figure 1: Architecture Overview

Extensions to the VIVO application and ontology form the basis for representing information about the content in the clearinghouse and for exposing that information through SPARQL[4] queries and other mechanisms to the rest of the components in the application.

Blacklight is a Ruby on Rails[5] gem that provides an interface to a Solr search index. The relationships expressed within VIVO are extracted using SPARQL queries to populate the Solr search index that enables discovery through the Blacklight front end. We extended Blacklight to

[1] http://www.vivoweb.org
[2] https://lucene.apache.org/solr/
[3] http://www.projectblacklight.org
[4] https://www.w3.org/TR/rdf-sparql-query/
[5] http://rubyonrails.org/

make requests to VIVO to extract information not conveyed through the Solr search index. One of the functionalities of the NYCCSC application is to enable searching using a map to find items in the clearinghouse that have been geotagged to locations within or close to the map bounding box. The application utilizes the Solr search index to store the geographical associations for VIVO entities which are represented by their URIs.

Sources

The VIVO project source code is available online. Instructions for installation can be found at https://wiki.duraspace.org/display/VIVO/VIVO+Technical+Documentation with specific instructions and documentation for version 1.8.1. at https://wiki.duraspace.org/display/VIVODOC18x/VIVO+1.8.x+Documentation. Downloading the 1.8.1 release (available at https://github.com/vivo-project/VIVO/releases/download/rel-1.8.1/vivo-rel-1.8.1.zip) provides all of the VIVO and Vitro source code required to install VIVO 1.8.1 .

The customized NYCCSC VIVO code is available at https://github.com/cul-it/nyccsc-vivo .
After downloading VIVO 1.8.1, you can use the following commands to clone the NYCCSC VIVO code repository in the directory of your choice (which we will call the NYCCSC VIVO installation directory):

- Git clone https://github.com/cul-it/nyccsc-vivo.git .
- OR ssh clone git@github.com:cul-it/nyccsc-vivo.git (if using SSH and with your SSH key and password already set on your computer).

The Blacklight project source code is available online at https://github.com/projectblacklight/blacklight.
The customized NYCCSC Blacklight code is available at the following GitHub repository https://github.com/cul-it/nyccsc-rails .

Installation: VIVO and SOLR

Application Setup and Deployment

The requirements for installation for the NYCCSC VIVO application are based on the requirements for installing the 1.8.1. VIVO application: you will need to install Ant[6], MySQL[7], JAVA, and Tomcat[8] to run the application. Please refer to https://wiki.duraspace.org/display/VIVODOC18x/A+simple+installation and review the

[6] http://ant.apache.org
[7] https://www.mysql.com
[8] https://tomcat.apache.org

“Preparing for VIVO - Install required software” section. At the end of this process, you should have a MySQL database defined for the application.

After cloning the base VIVO application and the NYCCSC VIVO application code using the GitHub links provided above, your directory structure should look like the following:

- VIVO 1.8.1 Installation Directory
    - Multiple folders including config, rdf, src etc. for 1.8.1
    - Vitro-core (contains the Vitro code used within the VIVO application)
- NYCCSC VIVO Installation Directory
    - Multiple folders including config, productMods, rdf, etc.
    - example.build.properties

Create a new directory for NYCCSC VIVO where some configuration and RDF data for the application will reside. We shall call this directory the NYCCSC VIVO home directory.

In the NYCCSC VIVO Installation Directory, copy example.build.properties to build.properties. You will need to set the following properties:

vitro.core.dir	path to Vitro directory within VIVO installation directory, e.g. /src/vivo-1.8.1/vitro-core
nihvivo.dir	path to VIVO installation directory e.g. /src/vivo-1.8.1
vitro.home	path to NYCCSV VIVO home directory e.g. /home/nyccsc_vivo
tomcat.home	path to Apache Tomcat’s base directory
webapp.name	the name of the webapp, e.g. nyccscvivo

Now, navigate to the NYCCSC VIVO installation directory and run ***“ant all”*** from the command line from within that directory.   If the build is successful, this step should create an example.runtime.properties file within this directory. Copy this file to runtime.properties and fill in the following properties with the correct values:

Vitro.defaultNamespace	http://www.nyclimatescience.org
rootUser.emailAddress	vivo_root@cornell.edu (replace with root email address)
VitroConnection.DataSource.url	jdbc:mysql://localhost/nyccscdb (the name should be the MySQL database created for NYCCSC)
VitroConnection.DataSource.username	nyccscdbusername (username for MySQL database)

VitroConnection.DataSource.password	nyccscdbpassword (password for MySQL database)
email.smtpHost	appsmtp.mail.cornell.edu (replace with smtpHost information for your server)
email.replyTo	vivoAdmin@cornell.edu
vitro.local.solr.url	http://localhost:7070/nyccscvivosolr (this is required to be the webapp name defined in build.properties followed by the word 'solr')

In the NYCCSC VIVO home directory, copy config/example.applicationSetup.n3 to config/applicationSetup.n3.

Duplicating the NYCCSC production site content and ontology

To get a copy of the content within the current NYCCSC website and ontology, you will have to copy over both the contents of the NYCCSC home directory as they exist in production as well as get a database dump from MySQL for the NYCCSC production database and import that into the database setup on your computer. The MySQL dump from the production database as well as a zipped copy of the RDF files can be provided for the purposes of replication of the entire production site if need be.

Given the MySQL dump from production, you would do the following in your local instance with an empty database. On the command line, replacing nyccscdbusername, nyccsdbpassword, and nyccscdb with your database's username, password, and name respectively, you can type: "mysql -u nyccscdbusername -p nyccscdbusername nyccscdb < mysqldump.sql" where mysqldump.sql is the MySQL database dump from the NYCCSC production site.

Build Procedure

As noted above, you should have already have run "ant all" which will create the appropriate webapp directory under Tomcat for deployment. At this point, you should have runtime.properties and applicationSetup.n3 defined in the appropriate locations (as described above). Your directory structure should look like the following:

- NYCCSC VIVO Installation Directory
    - Build.properties
    - Source code and other folders
- NYCCSC VIVO Home Directory
    - Runtime.properties
    - Config
        - ApplicationSetup.n3
- Base VIVO 1.8.1. Release
    - Vitro-core
    - Source code and other folders

After this process has completed successfully, start tomcat on your machine (e.g. sudo /etc/init.d/tomcat start).

Blacklight

Application Setup and Deployment

The NYCCSC Blacklight application runs using a Ruby on Rails platform and will require the installation and setup of Blacklight. Installation and setup documentation for Blacklight can be found at https://github.com/projectblacklight/blacklight/wiki/Quickstart .

Create a database in MySQL for the Blacklight application. This is a different database than the database for the VIVO application which was described above.

After cloning the NYCCSC Blacklight code to a directory which we shall call the NYCCSC Blacklight directory, you will need to do the following:

- Set environment variables for the following:
    -

'SOLR_URL'	this is the URL for the Solr search index. You can use the same local or fully qualified URL as specified in runtime.properties for your NYCCSC VIVO instance for vitro.local.solr.url.
'VIVO_APP_URL'	this is the URL for the base VIVO application
'VIVO_SPARQL_URL'	'VIVO_SPARQL_URL': this is the VIVO SPARQL URL defined in the section on installing VIVO
'VIVO_IMAGEURL'	This is the URL for the base VIVO application

- Update the following files:
    - *NYCCSC Blacklight directory/config/blacklight.yml*: Set the following property:
        - Set the appropriate environment url to ENV['SOLR_URL']. For example, if deploying Blacklight in production mode, write the following:
            - production : url: <%=ENV['SOLR_URL'] %>
    - *NYCCSC Blacklight directory/database.yml*:

        - Set database, username, and password information for the MySQL database to be used for Blacklight. This database is not the same as that used for the VIVO application. E.g.
            - Database: nyccsc_blacklight
            - Username: nyccsc_blacklight_username
            - Password: nyccsc_blacklight_password

- In the NYCCSC Blacklight directory, run the following commands if deploying this application in production for the first time:
    - RAILS_ENV=production bundle install
    - RAILS_ENV=production rake assets:precompile
    - RAILS_ENV=production rake db:migrate
    - Start rails server (if you need to manually start Rails with your Rails setup). The production server is running Passenger and requires a simple HTTP apache restart (e.g. sudo /etc/init.d/httpd restart) for the changes in the Rails application to be picked up.

Extensions, Customizations, and Ontology

This section provides an overview of the extensions made to the underlying VIVO code and Blacklight as well as an overview of the semantic approach used and ontology created for this project.

Extensions to and Customizations of VIVO

The table below summarizes some of the main customizations and extensions made to the VIVO application. The ontology section discusses the custom NYCCSC ontology and how it relates to the underlying VIVO ontology.

Search Index Population	Include additional SPARQL queries and logic to ● populate search index with both simple and hierarchical facet values (e.g. the sector facet and the climate changes facet) and ● Include additional fields expected by the Blacklight front-end ● to provide additional search result boosting for content types, such as maps, data, and documents, that need to be more visible in search results.
Search index configuration	● Update VIVO Solr schema to include additional fields to conform to the schema expected by Blacklight ● Within the VIVO package: ○ added libraries to be deployed with the Solr search index. ○ added synonyms that needed to be considered equivalent for discovery through the search index.
JSON Output of VIVO	Extend the profile output abilities to include JSON output of all the

Profile	properties displayed on the profile page. This JSON output is then used by the Blacklight front-end for display on the individual search result pages.
Custom forms for describing specific relationships	Include custom entry forms for adding and specifying highlighted content collections and the items within those collections.
Addition of reports SPARQL query pages	Using built-in VIVO page management functionality, include reports for content addition dates and other administrative information to help curation.
Image upload	Extended image upload functionality to include option for entering source and title for a particular image.
Geographical entities	Added RDF to describe NY state counties, towns, villages, and hamlets. These definitions enable these locations to be used in the geotagging of clearinghouse content.

Extensions/Customizations of Blacklight

The overall look and feel of Blacklight was customized based on user feedback and project team requirements. In addition to the visual and layout customizations, we made the following changes:

- Handling URIs as the main identifier: Customizations were required to ensure that URIs, which start with http://, would be handled correctly by Blacklight and Apache.
- Using cached SPARQL queries: Certain pages accessible through the site needed to incorporate SPARQL query results. We established a process whereby the SPARQL queries are defined as part of the Rails system, executed at predefined intervals or on system startup, and then cached for retrieval within the pages that rely on these results. VIVO's SPARQL Query API is used to execute the queries and the caching code is available through a separate gem for communication between VIVO and Ruby on Rails.
- Using direct JSON output from VIVO: We have kept the discovery layer focused on the content that needs to be searchable or displayed on the search results page. Since we also wanted to leverage the semantic relationships on an individual item or content page, we extended Blacklight to use the JSON output extension we added to VIVO to display additional information from VIVO on the details page for a search result. These extensions are available through a separate gem, titled "vivo-to-blacklight," for communication between VIVO and Ruby on Rails.
- Searching using the map: We incorporated a map viewer onto the search results page, enabling users to search for content geotagged by location using the map. We reviewed GeoBlacklight (http://geoblacklight.org) to see how they incorporated this functionality.

The NYCCSC Ontology and the semantic approach

Motivation and Driving Design Factors

Given the complexity of relationships between effects of climate change across sectors, specific vulnerabilities in those sectors resulting from climate change, and strategies to address climate change, we chose to adopt an approach for representing the content and related climate change concepts that could capture the richness and depth of these relationships while providing the flexibility to access information along different dimensions of interest. Using semantic relationships provides a richer model for content and supports more complex and involved queries as well as exposes additional information to the end- user. The semantic approach allows the application to use a common format to represent different types of data, such as different types of publications or documents, data products that present a view onto climate change data, and GIS layers representing geographical information relevant to climate change. From the curation perspective, this approach entails both thinking about the actual relationships between concepts and content but also being able to track content that has been entered and review existing concepts and relationships to understand how to categorize and link new content. The ontology has continued to evolve through the project lifecycle and the use of the semantic approach has provided us with the flexibility required for the iterative refinement of the concepts and relationships represented in the clearinghouse.

The custom ontology as it exists for this project models several climate change concepts, such as the effects of climate change and specific strategies for handling climate change, their relationships with each other, and their relationships to the different types of content that need to be discovered through the NYCCSC website. We designed the ontology to primarily meet the needs for discovery and access to content by the end-user. Since the NYCCSC VIVO instance serves as the main vehicle for content curation, we also had to consider how to support curation tasks. The following questions drove ontology design:

- End-User focus:
    - What relationships and information can help the end-user better understand the the effects of climate change that are relevant to their local community?
    - What are the relationships that can help the end-user in comprehending what plan of action or strategy they might employ?
- Curator focus:
    - What information would support curators in their review of content that has been entered and enabled curators to monitor the progress of the curation effort?

We used multiple sources of information to understand the main concepts we needed to represent, beginning with the ClimAID report which provides an overview of the effects of climate change within and across sectors. The ClimAID report provided detail on the changes that New York's climate has experienced and that are projected in the future, as well as the impacts of these changes and potential adaptation strategies. These details were incorporated into the ontology. In addition to the ClimAID report, we were guided to content through local governments and state organizations and programs such as the New York State Department of Environmental Conservation and its Climate Smart Communities program. We also find content from federal sources such as NOAA, EPA, and the U.S. Global Change Research Program.

Figure 2 shows an overview of how the ontology models climate change concepts and how they relate to each other, models the different categories of content made discoverable through

NYCCSC's search function, and links these concepts and content categories to actual maps, data, and documents. The discovery and curation processes are thus both enabled and supported by the ontology.

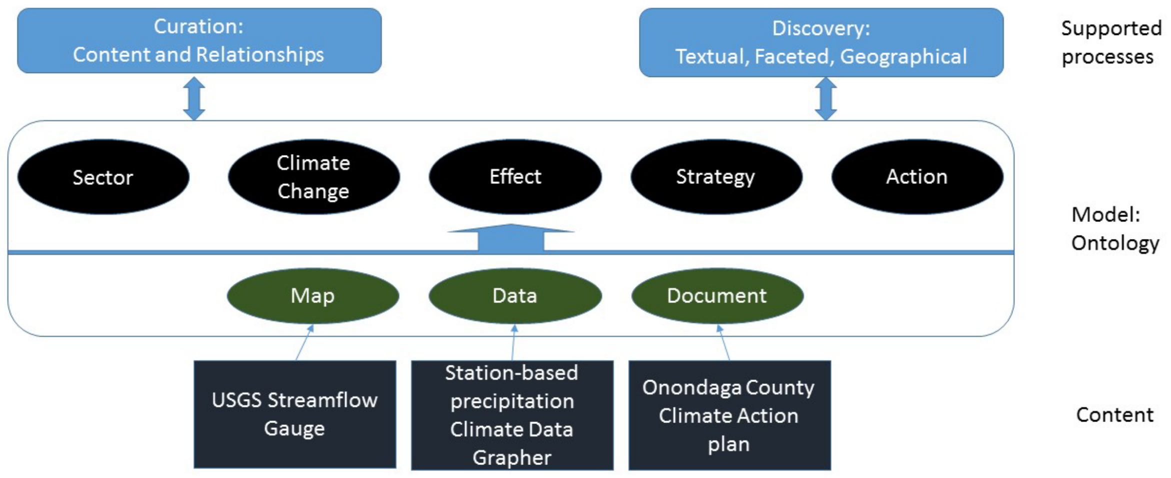

Curation:
Content and Relationships
Discovery:
Textual, Faceted, Geographical
Supported processes
Sector
Climate Change
Effect
Strategy
Action
Model: Ontology
Map
Data
Document
USGS Streamflow Gauge
Station-based precipitation Climate Data Grapher
Onondaga County Climate Action plan
Content

Figure 2: High level overview of ontology and its relationship to curation, discovery, and content

NYCCSC Ontology

Figure 3 shows a subset of the NYCCSC ontology and a part of the related underlying VIVO ontology. All nodes in the image are within the NYCCSC ontology namespace except for those with a prefix preceding the class name. Classes and properties included within the original VIVO ontology are shaded gray. Relationships with multiple intermediate nodes, such as "descendant" which indicates a class is a descendant but not a direct subclass of another class or the authorship relationship which utilizes additional relationships to link authors with information content entities, are indicated with dotted lines. Arrows point in the direction of the relationship Represented.

The VIVO ontology utilizes multiple ontologies and standards such as Friend of a Friend (FOAF)[9], the Bibliographic Ontology (BIBO)[10], classes compatible with the Open Biological and Biomedical Ontologies (OBO Foundry)[11], Simple Knowledge Organization System (SKOS)[12], VCard[13], and the Event[14] ontology. The NYCCSC ontology includes the use of the DCAT ontology[15] to specify dcat:Distribution as a superclass for Data

[9] http://www.foaf-project.org/
[10] http://bibliontology.com/
[11] http://www.obofoundry.org/
[12] http://www.w3.org/2004/02/skos/
[13] http://www.w3.org/TR/vcard-rdf/
[14] http://motools.sourceforge.net/event/event.html
[15] http://www.w3.org/TR/vocab-dcat/

Product. This relationship enables the use of dcat:downloadURL property to declare where the original data for a data product might be downloaded.

The ontology adds a GIS layer subclass to the OBO InformationContentEntity class from the original VIVO ontology. In the original VIVO ontology, Information Content Entity encompassed multiple subclass of BIBO Document and could be linked back to authors represented by FOAF Person. Utilizing the original VIVO ontology allows authorship information to be associated with data products, documents, and GIS layers in the clearinghouse.

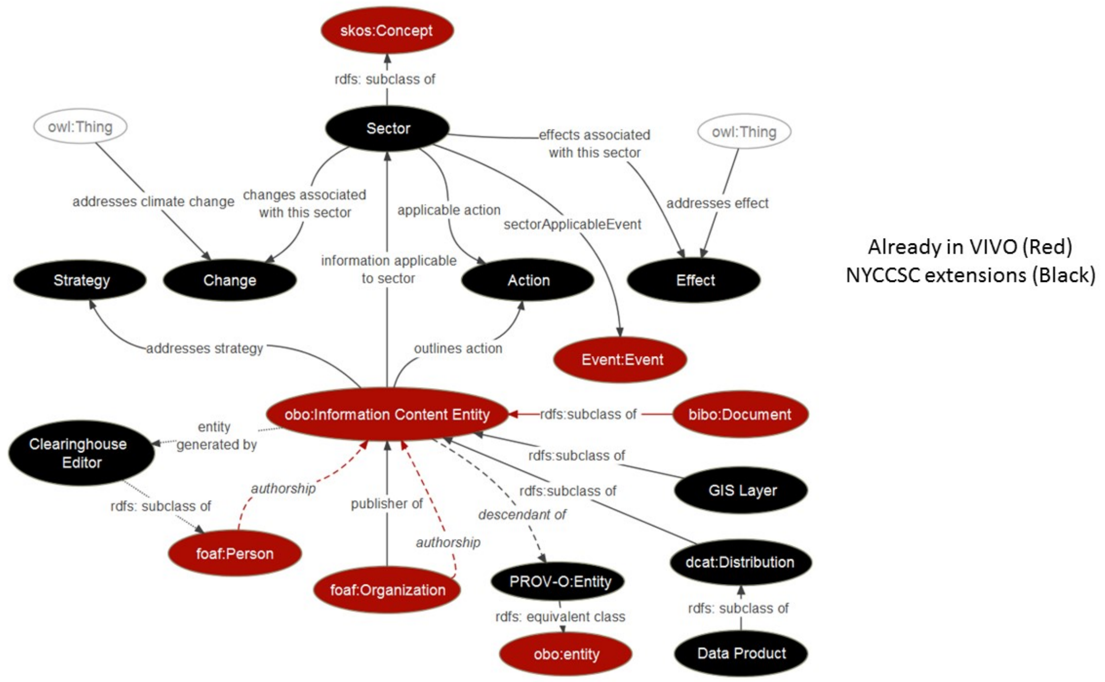


Figure 3: A high level overview of the NYCCSC ontology and how the ontology extends or uses existing classes within ontologies already in use within the base VIVO application

The main climate change concepts include Change which is a superclass for observed and projected climate changes, Effect which is further subclassed in two ways, into effects on organisms and systems and into specific vulnerabilities and opportunities that result from climate change, Action which has the subclasses Adaptation Action which represents mechanisms for addressing climate change adaptation needs and Mitigation Action which represents mitigation mechanisms, and Strategy (Adaptation or Mitigation). We chose to subclass Effect in two different ways in order to test two different approaches with users. A Sector is represented as a subclass of SKOS concept and links to Changes, Effects, and Actions. The “Addresses Climate Change” and “Addresses Effect” object properties have the respective ranges of Change and Effect but have no specified domain and are thus applicable to any entities defined in the system. The Information Content Entity class has multiple properties linking to Sector, Action, and Strategy.

The Clearinghouse Editor class is a subclass of FOAF Person and is used to represent individuals with curation or editing privileges. The VIVO application enables linking user accounts to specific instances of FOAF Person and can thus associate information about permissions and logins with particular people in the system. The ontology links the OBO Information Content Entity to the PROV-O Ontology[16] Entity by declaring that Information Content Entity's ancestor obo:entity is an equivalent class to PROV-O:entity. These linkages enable the use of the PROV relationship "this entity generated on" to track when a particular item was added to the clearinghouse.

The ontology also adds the relationship "this entity generated by" to track which individual instance of Clearinghouse Editor was responsible for adding the item to the clearinghouse. These classes and relationships thus serve to aid the curation effort.

Extending the underlying VIVO ontology has also provided us with the benefit of being able to expose linkages between content and people, organizations, programs and projects. For example, if the end-user sees a report published by the EPA Climate Ready Water Utilities program and they review information about the program itself, they will discover the program is also responsible for publishing a website and developing the Climate Resilience and Evaluation and Awareness software tool. VIVO's 'publisher of' property and authorship links can thus aid in discovery of additional relevant information. In addition, the NYCCSC ontology employs the Event ontology's Event class which is used within VIVO's ontology. Event instances can then be linked to sectors and tagged as addressing specific climate changes.

[16] http://www.w3.org/TR/prov-o/