VIVO v1.6 Release Notes

VIVO v1.6 Release Notes

What is this document?

The VIVO 1.6 Release Announcement covers the main features of the release at a high level but leaves out a number of helpful specifics. This document is intended to help those planning to implement the release as either a new installation or upgrade to learn more about new features and find pointers to more complete documentation elsewhere on this wiki.

The VIVO ontology has become VIVO-ISF

VIVO v1.6 incorporates major changes to the ontology that reflect a need dating back to 2009 to bring the VIVO and eagle-i project (https://www.eagle-i.net) ontologies closer together. NIH elected to fund two distinct projects focusing on the networking of researchers (VIVO) and research resources (eagle-i) but recognized the artificiality of the split and has encouraged collaboration, most recently by funding the CTSAconnect project (http://ctsaconnect.org) with Dr. Melissa Haendel of Oregon Health and Science University as principal investigator.

The new ontology is called VIVO-ISF, short for Integrated Semantic Framework. The full ISF ontology as published by the CTSAconnect project is freely available for download and includes technical documentation and its modules may be browsed online or downloaded. The VIVO 1.5 scope has informed selection of ISF modules appropriate for VIVO 1.6, with the anticipation that individual adopting sites may wish to include or exclude individual ISF modules based on their intended population of VIVO with data.

Documentation specific to VIVO-ISF lives on this wiki and will continue to evolve as the VIVO-ISF Ontology Working Group, led by Melissa Haendel and Brian Lowe, receives feedback from the rollout. Communities of interested are forming within the working group and will lead implementation of corrections, improvements, and expansion of the VIVO-ISF ontology for future releases. Please join the biweekly ontology working group calls.

To understand the VIVO-ISF changes from previous releases, we call particular attention to the VIVO 1.5 to VIVO-ISF change diagrams. Upgrading VIVO from version 1.5.x to version 1.6 includes an automated data migration process that should convert all existing VIVO content to the new ontology structure, but any local data ingest processes will almost certainly need to be modified to generate data compatible with the new ontology. We recommend backing up the VIVO 1.5 database and performing a provisional upgrade on a test server to confirm that the upgrade will complete smoothly – please see the VIVO 1.6 upgrade documentation for more details.

Internationalization

VIVO has gained significant adoption outside North America and internationalization of the application menus and other interface elements and ontology labels has emerged as a high priority, sometimes even in other English-speaking countries where even minor terminology tweaks have until now required modifying the VIVO application. VIVO v1.6 leverages Java resource bundles to separate all text strings embedded in menus, page templates, dialog boxes, error messages, and ontology class and property labels from the application itself so that translations can be accomplished without programming. Separating language-specific text from the code also minimizes the work of updating alternate language versions of VIVO for future releases; a comparison of the changes in the English text from one version to the next can pinpoint areas where text in another language would also need modification.

We characterize the internationalization in version 1.6 as *read-only* because significant additional work will be required to make interactive editing of all VIVO content in multiple languages possible. Content in the form of RDF bearing standard language tags and imported into VIVO will be displayed appropriately; VIVO will respect a user's current browser-based language preference settings to find the closest match based on BCP 47 standard language tags – for details see the W3C document on setting language preferences in a browser.

VIVO v1.6 does support editing each individual entity's primary label to supplement the default language with additional labels in other languages. While limited to labels, this would allow showing multiple versions of a person's or organization's name as well as including titles of books or other publications in multiple languages. Users whose browser settings indicate a language preference will see the appropriate language by default, with an icon to indicate that additional languages are available for that individual entity.

VIVO v1.6 still ships with U.S. English as the default language; a test bundle prepared using Google Translate for testing only is available and we expect an announcement soon of an contributed version prepared by VIVO community members.

Web services

VIVO 1.6 introduces important new functionality in the form of a web service designed to accept SPARQL update commands authenticated via username and password. One primary motivation for adding a write capability is to ingest data into VIVO that triggers updates to the search index as well as performing any needed re-inferencing. The VIVO Harvester has previously been able to write to the VIVO database directly by issuing Jena library calls, but using the VIVO web service will avoid the extra step of rebuilding the VIVO search index and re-computing inferences after each Harvester run.

The web service implements SPARQL 1.1 Update to allow other applications to write to VIVO via a well-documented standard. The write capability of the VIVO web service can be used in conjunction with linked data requests, VIVO's internal SPARQL query interface, or a SPARQL endpoint such as Fuseki to accomplish round-tripping of VIVO data in other applications ranging from lightweight JavaScript visualizations tools to full-scale web applications. Several VIVO community members leverage the native RDF capabilities of the Drupal open-source web content management system to create dashboards that re-purpose VIVO data, and with the new web service these independent applications will be able to create, update, and delete as well as read VIVO data.

Chris Barnes from the University of Florida is leading a new VIVO Apps & Tools working group showcasing existing tools via demos and discussions on biweekly calls, as well as addressing opportunities and requirements for new tools and applications beneficial to the VIVO community.

Performance improvements

VIVO draws the content displayed on a typical profile page from a very disaggregated structure represented internally as RDF statements, or triples. Hundreds or even thousands of RDF statements are dynamically assembled or 'rendered' into HTML for display of a page; the process naturally takes longer as the amount of information to display on the page increases.

Applications that generate pages from underlying databases can often take advantage of HTML caching – a technique to store a copy of the finished page to be served to the user instead of regenerating the page from scratch at every request. There are a number of standard HTML caching libraries that leverage the hypertext transfer protocol (HTTP) headers returned in response to an incoming browser request to permit an application to only regenerate a page when its content has changed. VIVO 1.6 has been modified to use a field in the Apache Solr search index that reflects when a page has last been edited as a trigger to regenerate the page; if the cached version of the page is still newer than the date and time of last update, the page stored by the caching library is returned immediately, reducing server load and speeding the response to the user. Note, however, that caching is disabled for a user whenever that user is logged in.

Mark Fallu from Griffith University and Ted Lawless from Brown each contributed to this new functionality.

VIVO's search indexing utility has also been extended to support re-indexing a specific subset of data known to have changed. If through data ingest or other modification a known set of individual entities has been updated, VIVO's search indexing can be given a list of URIs to index rather than having to re-index the entire database. This should allow more efficient processing of incremental updates to publications or other content in batch mode, especially when using the VIVO Harvester and writing directly to the VIVO database using the Apache Jena code libraries.

In other performance-related improvements, responses to linked data requests are more concise, hence faster, and include a link to the VIVO "terms of use" statement. The terms of use text should be modified by the implementing institution to clarify any local conditions on usage of the information in VIVO that is included in response as determined by visibility settings for page display in VIVO.

Look and feel

VIVO keeps the same overall look and feel while featuring a new and more dynamic home page including rotating features highlighting individual research areas, researchers, and departments as well as more prominent statistics on key content elements. An optional map view highlighting the global, national, or regional geographic research focus may also be activated and all new home page features may be customized to local preference.

And many more improvements

In addition to the above major features, VIVO 1.6 includes many new development and debugging features offering implementing sites additional control over deployment, access control, and customization of VIVO pages with additional queries and reports.

Jim Blake has implemented a developer mode that can be configured on startup and modified while running, having several notable features:

- Adding HTML comments to each Freemarker template, so you can see what each template contributes to the page by viewing the source of the page in the browser
- Defeat the Freemarker template cache, so each template is read from disk on each request
- Turn on logging of custom list view configuration files to note in the log each time a list view other than the default is used
- Defeat the cache of language-specific text strings, so the language file is read from disk on each request
- Turn on logging of all SPARQL queries to include the elapsed time spent on the query, in seconds, the name of the method on RDFService that received the query, the format of the result stream from the RDFService method, and the text of the query. A stack trace may optionally also be added to the log.

VIVO's internal SPARQL query endpoint may be configured for authorized access and now supports HTTP content negotiation and JSON-LD.