VIVO v1.6 release planning

Daily release planning standup meetings are held at noon Eastern Daylight Time

- Track progress on current issues
- VIVO 1.6 release announcement
- VIVO v1.6 Release Notes

Why v1.6 and not 2.0?

Given the ontology and directory layout changes, **should this be version 2**?

- Pros: this is a bigger jump than a normal point release, especially with the ontology and data migration, but also with the RDF initialization files rearrangement
  - people will need to be aware that while the upgrade process will migrate existing data, their application customizations and data ingest scripts will very likely need to be modified
  - jumping to 2.0 signals a bigger change
  - we can go to 2.1 if there bug fixes and/or ontology revisions need to happen soon

- Cons:
  - several of the key features for 1.6 are in their first implementation state and not fully mature
    - e.g., multiple language support will only support entering content in multiple language for rdfs:labels, not for data properties or custom forms
    - the web service for adding and updating RDF via SPARQL update will likely want to be extended to support queries (currently reads are accomplished through linked data requests, by changing permissions on VIVO's embedded SPARQL query page, or by a sepa
      - Unable to locate Jira server for this macro. It may be due to Application Link configuration.
  - other features we
    - updating the Jena libraries, which will require removing dependence on the Jena RDB triple store technolog, still currently used for user accounts and other internal application data
    - external concept linking to Library of Congress Subject Headings and the National Agriculture Library Thesaurus
    - being able to select people from another VIVO
    - addressing organization identifiers
    - other post-1.6 issues

- Road map consideration – the 2.0 release would serve as an excellent driver for road map discussions defining goals, features priorities, and resource requirements with 2.0 a near enough milestone that division of tasks between 2.0 and post 2.0 could be effectively addressed

*Please note that these are **proposed** features for VIVO 1.6, not commitments by the VIVO development community*

- Performance
  - page caching
  - other approaches
- Installation and testing
- Adapting to the ISF ontology
- Site and page management
- Support for sameAs
- Web service for the RDF API
- Serialize/restore graphs in quad format
- Editing
- Ingest tools
- Internationalization
- Provenance
- Visualization
- Data query, export and reporting
- CV generation
- Search indexing
- Modularity
- Ontology
- Tools outside VIVO

Background

The VIVO 1.6 development time frame is roughly November through June with the goal of having a VIVO 1.6 out before the VIVO Conference, to be held this year in St. Louis, Missouri from August 14-16.

The list of candidate features and tasks below include both major and minor items from earlier road maps and development planning documents. Where appropriate, stages are suggested for development and implementation to reflect necessary design time or to allow for refinement of an initial design based on user feedback.

Please also note that significant changes are being made to the VIVO ontology through the CTSAconnect project and collaborations with euroCRIS and CASRAI. We anticipate more changes to the ontology for VIVO 1.6 than in recent releases, but most of the changes will affect the modular organization of the ontology and the class and property URIs and/or labels rather than the structure or design patterns in the ontology.

The following **proposed** features are not in priority or temporal order.

Key to symbols

✅ the work for this is basically done, even if an issue has not yet been created to track final completion

👍 we have this work underway – check the related JIRA issue to track progress

❓ we think we understand this but are not sure it will make it into the release – if there's a JIRA issue linked, please go vote on it and comment to indicate why it should be either bumped up or deferred

💡 good idea that we aren't quite sure what to do with yet

⚠️ this feature still needs further clarification, requirements analysis or design work so will not be included in VIVO v1.6

👎 we don't see any way to do this for 1.6

Performance

There are a number of possible routes to performance improvement for VIVO, and we seek input from the community on what the primary pain points are. Some performance issues are related to installation and configuration of VIVO and we are working on improving documentation, notably on MySQL tuning, and troubleshooting, but page caching has emerged as the primary performance-related improvement for 1.6.

Page caching

✅ Unable to locate Jira server for this macro. It may be due to Application Link configuration.

As more data is added to VIVO, some profiles get very large, most commonly when a person accumulates hundreds of publications that much each be included in rendering the person's profile page. While we continue to look at ways to improve query speeds, if you need your VIVO to display all pages with more or less equivalent, sub-second rendering times, some form of page caching at the Apache level using a tool such as Squid is necessary. Apache is very fast at displaying static HTML pages, and Squid saves a copy of every page rendered in a cache from which the page can be rendered by Apache rather than generated once again by VIVO. The good news:

- VIVO's Solr (search) index has a separate field for each Solr document indicating the date and time that VIVO page was last indexed following a change; a Solr document corresponds to a page in VIVO.
- If we make this last update field available as part of the HTTP header's for a VIVO page, Apache and Squid can compare that datetime value to the datetime value in the Squid cache, and only re-generate the page from VIVO when the cached version has been superseded by changes in VIVO.

However, there are several issues to be resolved in making this work as anticipated

- We need to verify that the field holding the date and time of last update in Solr does indeed reflect all the changes that we think should be reflected.
- Pages in VIVO may have many different incarnations depending on whether a user is logged in or not, and if logged in, depending on their level of editing privileges. The caching solution being implemented for 1.6 will disable use of the cached pages whenever a user is logged in to edit, even on pages where they have no editing rights.

Griffith University has implemented page caching and Arve Solland gave a talk on this and other aspects of the Griffith Research Hub at the 2012 VIVO conference.

Other approaches to performance improvement

There are also other ways to address performance that could be argued are more effective in the long run

- ✅ As mentioned above, improved server, Apache, Tomcat, and database configuration and tuning
- ⚠️ (not part of the 1.6 release – more requirements needed) If we can identify key areas where some form of intermediate results are being repeatedly requested from the database, implementing Memcached could be another strategy. However, it may be more effective to provide MySQL more memory since it can use its own strategies for query caching
- ✅ Tim Worrall has been looking at our page templates for instances where we could avoid issuing SPARQL queries for the same data repeatedly in the course of generating a single page, and has also been optimizing SPARQL queries that come to his attention
- ⚠️ (not part of the 1.6 release – independent investigation) There is also some indication that bugs in Jena's SDB implementation that make queries other than to a single graph or the union of all graphs much less efficient, at least for MySQL. This is hard to verify, and we have mostly been approaching this by exploring the use of other triple stores via the RDF API added with the VIVO 1.5x releases.

Installation and Testing

- Brian Caruso has proposed adding a unit test for Solr that would build an index from a standard set of VIVO RDF, start Solr, and run standard searches. This would help prevent breaking existing functionality when addressing issues that have come up such as support for diacritics, stop words, and capital letters in the middle of names
  - ❓ A unit test has been developed for another related project at Cornell and we hope to be able to port this to VIVO, but perhaps not for
  - Unable to locate Jira server for this macro. It may be due to Application Link configuration.
- ⚠️ (not fo hat performance is highly installation dependent. The most urgent problem at Cornell has been the intermittent loss of communication between the VIVO web server and

the VIVO database server, which results in some threads of activity simply hanging and never returning. As with many errors that are hard to reproduce, we have developed workarounds that divide large jobs into chunks of data that experience has shown can be removed or added without causing hiccups.
    - Joe M. has submitted a paper to the Conference on a data ingest method using a standard set of data. This could conceivably be extended to serve as a set of tests, but is presently more geared toward helping people new to VIVO understand data ingest than testing VIVO under load

Adapting VIVO to the Integrated Semantic Framework ontology

👍 Unable to locate Jira server for this macro. It may be due to Application Link configuration.

Site and Page Management

- Make the Ab
    - 👍 Unable to locate Jira server for this macro. It may be due to Application Link configuration.
    - Not
- ✅ Offering areas, international focus, or most rec
    - Unable to locate Jira server for this macro. It may be due to Application Link configuration.
- ✅ Offeri
    - Unable to locate Jira server for this macro. It may be due to Application Link configuration.
    -
        - could put the service-specific aspects in a sub-template that gets imported and could be default not attempt to capture and cache images at all
        - are free services out there, but they may not be there in 6 months
- ✅ Offering the ability to embed SPARQL query results in individual pages on a per-class basis – for example, to show all research areas represen
    - Unable to locate Jira server for this macro. It may be due to Application Link configuration.
    - re broadly than for academic departments, for example
        - And a concept doesn't have the reverse of that, so from a concept you can see the departments involved and not just the people – Stephen will be building that
        - Jim – the data getter is specified in an RDF file that is loaded when the application is started up, saying that a class has a data getter in it
            - Stephen – is there a way to make that a list?
    - Jim – have wanted to make this part of the application configuration ontology
    - Stephen – is developing visualizations around shared research interests

Support for sameAs statements

background

When 2 URIs are declared to be the same in VIVO, all the statements about both will be displayed for either (e.g., Israel and Israel).

- Colorado has a use case to assert sameAs relationships between people's profiles in their university-wide VIVO and the separate implementation at the Laboratory for Atmospheric and Space Physics, where additional information about research projects, equipment, and facilities will be stored behind a firewall. They would like to pull data from the CU VIVO into the LASP VIVO dynamically, and pull any publicly visible data from LASP VIVO to supplement the CU VIVO content about a person.
- Colorado also has a need to pull data from the Harvard Profiles system used by the University of Colorado Medical School in Denver to the CU VIVO without replicating more than is necessary. This is a similar use case to the connections needed between VIVO on the Cornell Ithaca campus and the Weill Cornell VIVO.

Improvements are needed, however:

For 1.6

It may be a first step to simply show a link to the equivalent URI, with some form of "see also" label

A constrained approach could disable sameAs reasoning in our simple reasoner, which will start filling in for one URI everything that is inferred about the other

- add the switch to configuration
    - 👍 Unable to locate Jira server for this macro. It may be due to Application Link configuration.
- offer it as a s and to display the statements
    - 👍 Unable to locate Jira server for this macro. It may be due to Application Link configuration.

More general support for sameAs

- It will adversely affect performance to require VIVO to detect that two or more URIs for an individual have been declared to be sameAs each other and then retrieve and blend all the data for each URI for rendering on a single page
    - This becomes more complicated when the 2nd or higher URI is not in the local VIVO
- sameAs in the object position
    - when another VIVO individual is linked to one or the other of these Israels, the application is not yet smart enough to show only one object property statement to one instance of the Israel, and it looks to users like a duplicate of both the country and the relationship

Web service for the RDF API

- 👍 Unable to locate Jira server for this macro. It may be due to Application Link configuration.
- Imp her tools to add/remove data from VIVO and trigger appropriate search indexing and recomputing of inferences.
    - This would also enable round-trip editing of VIVO content from Drupal or another tool external to VIVO via the SPARQL update capability of the RDF api
    - Authentication will be involved
        - Could manage in our own authentication and authorization system and tell Apache that the servlet requires an HTTPS connection
        - This approach would allow testing in a known environment without having to set up SSL certificates
    - It would help the user experience if it's possible to bundle together an atomic change set (at least all those changes for one graph), so additions and retractions would not show up piecemeal
        - Note that since inferences are done is a separate thread there may still be some lag
- ⚠️ Put and delete of data via LOD requests – this has been suggested but we're not sure a specification even exists for an LOD "put" request – please add references here if you're aware of discussion or documentation.
    - when we were design the RDF API, if we allow anybody to execute an arbitrary SPARQL update or delete, you can't just listen to the changes, so we limited what we supported through the RDF API to adds or deletes, using just a subset of the overall language
    - the idea of being able to pipe an arbitrary update or delete through our API would take some work but is theoretically possible
- Stephen is willing to test for the Harvester

Serialize/restore a set of graphs in quad format

Unable to locate Jira server for this macro. It may be due to Application Link configuration.

This would support reading and writing data like class groups

Editing

- ⚠️ Being able to configure the relationships that determine what is editable through self-editing, such as that an author of a paper can edit it while nobody else can. The caution is that this has a complexity similar to a generic deletion problem – how far does the authority bridge?
    - See comments on VIVOIMPL-15 with respect to improving the permissions scheme for editing and make its functions more transparent to users. The best way forward here would be to transfer what's referred to as the editing policies, now hard-wired in code, to a set of RDF statements conforming to an editing policy ontology and editable from the Site Admin menu. This was the approach taken for the user model, and is proposed as an improved way of managing the display model (primarily for managing menu pages) and the application configuration ontology.
- Improve editing of data held in context nodes from the organization, event, or other related entity, principally via relationships like authorships or positions or via roles realized in processes or events – most custom forms support entry and editing only from the person. This requires no new functionality but will involve implementing additional custom forms

Other candidate issues relating to content editing

- NIHVIVO-1126 support for editing rdf:type of individuals via primary editing interface, not just the admin editing interface, as requested by Brown to allow one type of Document to be changed to another when the original automated type assignment was inappropriate.
    - Question – is this a functionality for any self-editor or just for privileged editors (e.g., library curators)?
    - The question has come up here about whether to remove statements that are no longer appropriate based on the domain of the property with respect to the new rdf:type.
        - We think not – the VIVO rendering interface will continue to show the statements, and existing statements will be editable
        - If the user removes a statement, the option to add a new statement may not be offered. This is appropriate.
        - If the user changes his or her mind and changes back the rdf:type of the individual, the original statements would still be there
- NIHVIVO-1125 class-specific forms for entering new individuals
    - These can be implemented as needed without new code functionality, although the custom forms would require adding the appropriate editing configurations to the code
- The first parts of the application configuration ontology, such as treating the same property differently based on the type of the related entity
    - what label, but also more
    - what property group
    - what editing form

Ingest Tools

- 💡 Integrating Mummi Thorisson's Ruby-based CrossRef lookup tool for searching and loading publications into VIVO, on GitHub along with OAuth work for retrieving information from a VIVO profile in another application
- 💡 Improving and documenting the Harvester scoring and matching functions
    - ✅ Stephen is already working on updating the Harvester documents in general – not sure how far into the scoring and matching documentation he will get

- ✅ Look for a poster by Joe McEnerney detailing a worked ingest example

Internationalization

👍 Also referred to and documented as Multiple Language Support in VIVO

- We are moving text strings from controllers and templates to Java resource bundles so that other languages can be substituted for English
- We will support internationalization for ontology labels – important because much of the text on a VIVO page comes directly from the ontology
- We are translating text in JavaScript and in properties files
- There are many places to track down in page management, the index page controller, etc., but that process is going well
- The Site Admin page will be translatable
    - but JSP pages left in the back end editing and supporting some ingest functions will not be translated, in part because we need to consider whether to put any available effort into moving them to FreeMarker as a higher priority, or perhaps at the same time as a way to kill two birds with one stone

👎 Improving the VIVO editing interface(s) to support specification of language tags

- VIVO 1.5 will respect a user's browser language preference setting and filter labels and data property text strings to only display values matching that language setting whenever versions in multiple languages are available – but there has not yet been a way to specify language tags on text strings.
- This affects the default data property editing form, editing multiple rdfs:labels, and any custom forms that add
    - we need to design the UI
        - if the editing user has Spanish as the preferred language, do you assume all text entered is in Spanish?
        - if more than one version of a string or label is present in the data, do you offer both for editing?
    - do we need a batch process to convert existing data to add language tags based on what is stated to be the default language of that VIVO installation?

Provenance

Adding better support for named graphs in the UI (the application already handles named graphs internally and through the Ingest Tools menu).

In many cases one goal of using a named graph is to assure that the content in the graph is **not** editable, so we need to be careful here. For 1.6, the enhancements under consideration include

- ❓ Allowing the addition of statements about any named graph such as its source and date of last update
- ❓ Making this information visible in the UI (e.g., on mousing over any statement) to inform users of the source and date of any statement, at least for data imported from systems of record

Visualization

- 👍 improved caching of visualization data – a student project at Indiana University has investigated and traced the issue as a problem in allowing multiple concurrent threads trying to create the cache of data for the same type of visualization.
    - This has been reported instead as a problem with scalability (e.g., for a Map of Science from the 32,000 plus publications in the University of Florida VIVO, or at UPenn)
    - This may solve the problem and will at least make it easier to determine whether further work on caching is necessary – if so, a solution for caching intermediate data vs. the final resulting page or image is likely to make sense
    - Jim will finish processing the pull request from Indiana that fixes a concurrency bug that might start more than one instance of a long process
- 👎 HTML5 (phasing out Flash) – not likely to be addressed in 1.6
- Map display on the home page
    - 👍 a static query for that map – only queries geographic regions; the ability to do countries, states and/or counties is anticipated but currently broken
    - get that fixed so could turn those on and off through modification of the Freemarker, or ideally through an admin interface
    - Stephen will create a blocker issue in JIRA

- ❓ Adapting the visualization to work for shared research areas (Colorado interest) and/or shared presentations (Brown interest)

Data Query, Export and Reporting

- 👎 Limiting SPARQL queries by named graph, either via inclusion or exclusion.
    - This is allegedly supported by the Virtuoso triple store. This would help assure that private or semi-private data in a VIVO could be exposed in via a SPARQL endpoint
    - If this functionality is dependent on the underlying triple store chosen for VIVO, it's not something that can easily be managed in VIVO
- There are other possible routes for extracting data from VIVO including linked data requests – if private data is included in a VIVO, all query and export paths would also have to be locked down. Linked data requests respect the visibility level settings set on properties to govern public display, but separate more restrictive controls may be required for linked data.
    - (Eliza) can't
- ❓ Being able to get the linked data for a page as N3, Turtle, or JSON LD, not just RDF/XML
    - would be trivial – N3 and RDF/XML are now supported; Turtle is just a subset of N3 and we're not using any of the non-Turtle features of N3
    - VIVO doesn't currently support ntriples, another subset of N3 that lists each triple
    - not sure of the JSON-LD – raised by Eric Meeks
        - Ted – there's some indication that a JSON-LD serializer for Jena is being written-

- Enhancing the internal VIVO SPARQL interface to support add and delete functions, not just select and construct queries – see "Web Service for the RDF API" above

CV generation

- Improving the execution speed and formatting of the existing Digital Vita CV tool as implemented in the UF VIVO, perhaps changing it to email the generated CV as a rich text or PDF document asynchronously
  - The latter is the more promising approach – if people don't have to wait but can have the document emailed to them, then the perception of slow performance is largely moot
  - That said, it may be possible to improve the queries. Florida and Stony Brook are known to be using this functionality so should be involved in prioritizing any changes
- Developing a UI for selecting publications or grants and adding required narrative elements, based on the specification developed in the Digital Vita VIVO mini-grant
  - This has been on "the list" for two years, but makes most sense as a different application
  - Many people have voiced the opinion that getting the data out in an form that is editable in Word or other common document editing tools is much more important than managing the selection or ordering of content from within VIVO

Search indexing improvements

- Provide a way to re-index by graph or for a list of URIs, to allow partial re-indexing following data ingest as opposed to requiring a complete re-index
  - Unable to locate Jira server for this macro. It may be due to Application Link configuration.
- The sam
  - Unable to locate Jira server for this macro. It may be due to Application Link configuration.
  - the delta – what has been removed as well as what has been added – and this may be more complex than search re-indexing a specific set of URIs
- Implementation of additional facets on a per-classgroup basis – appropriate facets beyond ref:type, varying based on the nature of the properties typically present in search results of a given type such as people, organizations, publications, research resources, or events.
  - Huda Khan has been implementing the ability to configure additional search facets for the Datastar project; some improvements may make it into 1.6
- An improved configuration tool for specifying parameters to VIVO's search indexing and query parsing
  - Question – are any of these run-time parameters or are they all parameters that must be baked in at build time, requiring re-generation of the index?
  - Relates to another suggestion for a concerted effort to explore what search improvements Apache Solr can support and recommendations on which to consider implementing in what order
  - Changes are not expected for 1.6 – more requirements are needed before this work can be prioritized or scoped.
- Improved default boosting parameters for people, organizations, and other common priority items
  - Here the question immediately becomes "improved according to what criteria"
  - This is a prime area for a special interest group of librarians or other content experts willing to document current settings and recommend improvements, including documenting use cases and developing sample data that could be part of the Solr unit tests listed above under "Installation and Testing"
- Improving the efficiency and hence speed of search indexing in general – we have no indications at the moment that search indexing is being a bottleneck. It can take several hours to completely reindex a major VIVO such as Florida or Cornell, but the ability to specify a single named graph or list of URI's to index would address most of the complaints around the time required search indexing after adding new data via the Harvester, which does not trigger VIVO's search indexing or re-inferencing listeners

Modularity

Jim Blake did significant work during the 1.5 development cycle learning about and the OSGi framework and exploring how it could be applied to VIVO, as documented at Modularity/extension prep - development component for v1.5.

- Yin's alternate search approach at NYU that indexes everything in the context of connections to people and displays results only for people could be of interest to others but would require modularity in search indexing code as well as other ways that the search index integrates with VIVO

There are no plans to implement additional modularity inside VIVO for 1.6, although the Web service for the RDF API work element would enable other applications to write as well as read VIVO data and support a more modular approach to adding functionality to VIVO.

Tools outside VIVO (not linked to the 1.6 release)

Weill's VIVO Dashboard

Paul Albert has been working with a summer intern and others at Weill Cornell to develop the Drupal-based tool for visualizing semantic data. This project provides a number of candidate visualizations and reports that will likely be of interest to other VIVO adopters, and there may be enhancements to VIVO that make this kind of reporting dashboard easier to implement.

URI Tool

The URI Tool is a separate, simple application designed to facilitate data cleanup in VIVO following ingest, often from multiple sources. The tool can be configured to run one of four or five pre-defined queries to identify journals, people, organizations, or articles with very similar names. A bare-bones editing

interface allows a relatively untrained user to step through lists of paired or grouped candidates for merging, identify which existing properties to keep, and confirm that the candidates should be merged. Links to the actual entry in VIVO facilitate verification. When the review process is complete, the URI Tool application writes out both retraction and addition files, which can then be removed from or added to VIVO using commands on the ingest menu.

This tool does not replace the need for author disambiguation and other cleanup work prior to ingest, for which the Google Refine extensions for VIVO and the Harvester tool have been developed. However, it does have the potential to become a considerable time saver for cleaning data once loaded into VIVO.

- further generalization and documentation of Joe McEnerney's URITool, including support for finding and removing/correcting data in only one named graph
- improvements to the interface