Upgrade Guide v1 to v1.2

VIVO Release 1 v1.2 Upgrade Guide

February 16,2011 - Upgrading from Release 1 v1.1 to Release 1 v1.2
*Release announcement for V1.2
*Upgrade process for V1.2
This document provides a short description of the steps involved in upgrading your installation of VIVO from Release 1, Version 1.1 to Version 1.2. This and other documentation can be found on the support page at VIVOweb.org

If you need to do a fresh install, please consult the VIVO Release 1 v1.2 Installation Guide found on vivoweb.org or the install.html file located in the doc directory of the VIVO source code distribution. The installation document also has a list of the required software and versions.

Release announcement for V1.2

The VIVO 1.2 release incorporates major changes throughout the application - notably a new templating system to support more versatile page rendering, plus improvements to address scalability. The release also features a new personal visualization option covering grants as well as publications. The VIVO Harvester library has also been significantly improved and expanded in scope for its 1.0 release through the VIVO SourceForge project at http://sourceforge.net/projects/vivo.

Templating system for page generation, navigation, and theming

A fresh installation of VIVO 1.2 looks strikingly different, with the introduction of a new default theme which takes advantage of the navigation and browse features delivered by the templating system. Individual pages now offer inline navigation to streamline viewing of expanded personal and organizational profiles, as well as improved content layout and organization. New browse controls on the home page and menu pages help to provide an immediate overview of the size and range of content and quick access down to the individual person, organization, research feature, or event.

Storage model

While server memory capacity has increased significantly in recent years, VIVO's reliance on in-memory caching of RDF data had put limits on the ultimate scalability of VIVO instances and potentially increased the cost of servers required to support VIVO.

With version 1.2, VIVO has been converted to optionally use Jena's SPARQL database (SDB) subsystem. SDB significantly reduces the baseline memory footprint, allowing VIVO installations to scale well beyond what has previously been possible.

New visualizations

Visualizations of networks of co-authors are now complemented by visualizations of co-investigators on grants, with similar interactivity and options for export as images or data.

Ontology

VIVO 1.2 includes a new ontology module representing research resources including biological specimens, human studies, instruments, organisms, protocols, reagents, and research opportunities. This module is aligned with the top-level ontology classes and properties from the NIH-funded eagle-i Project.

Associated VIVO releases

VIVO Harvester

The Harvester development team is releasing version 1.0 of the VIVO Harvester library shortly following the release of VIVO 1.2. The Harvester is an extensible data ingest and updating framework with sample configurations for loading PubMed publication, grants, and human resources data. Pre-release versions of the Harvester are available at http://sourceforge.net/projects/vivo.

Upgrade process for V1.2

Before Performing the Upgrade

Please ensure that backups are created of the:
*Tomcat webapps directory
*Original source directory
*MySQL database (mysqldump)
The upgrade processes similar to the original install process with the following EXCEPTIONS:
*DO NOT reinstall MySQL or recreate the MySQL database. Please ensure that you back-up the MySQL database. Also note that VIVO 1.2 will not run on older versions of MySQL that may have worked with 1.1.1. Be sure to run VIVO 1.2 with MySQL 5.1 or higher. Using unsupported versions may result in strange error messages related to table formatting or other unexpected problems.
*It is not necessary to add RDF data.
*First-time login of the administrator account after the upgrade process is complete will use the password previously set, NOT the default password used on the first login after the initial installation.
*The first time Apache Tomcat starts up after the upgrade, it will initiate a process that modifies the knowledge base to align the data with the revised ontology. See the section on the Ontology Upgrade below for more information.

Choose Triple Store

VIVO 1.2 offers a choice of two triple store technologies: in-memory models backed by Jena's legacy relational database store (RDB), and Jena's SPARQL database (SDB). RDB was used by VIVO 1.1.1 and earlier. This mode offers fast response, but only by caching the entire RDF model in the server's main memory. The memory available to VIVO limits the number of RDF statements that may be stored.

SDB mode caches only a fraction of the RDF data in memory. Most queries are issued directly against the underlying database. This allows VIVO installations to display data from large RDF models while requiring only a small amount of server memory to run the application. There is a tradeoff in response time: pages make take slightly longer to load in SDB mode, and performance will depend on the configuration parameters of the database server. Additionally, advanced OWL reasoning (not enabled by default in either mode) is not possible in SDB mode. With SDB, only the default set of inferences (inferred rdf:type statements) are generated, though they are generated as soon as data is edited rather than in a background process.

Though a VIVO installation may be switched back and forth between RDB and SDB mode by changing a configuration property and redeploying the application, it is important to note that data added in one mode will not typically appear in the other. The exception is when a system is first switched from RDB mode to SDB mode. In this case, the data from the RDB store will be automatically migrated to SDB.

A VIVO 1.2 system that is upgraded from VIVO 1.1.1 must initially be run in RDB mode in order to receive required ontology updates. Attempting to run an upgraded system initially in SDB will result in a logged error message, and the application will not start. After the system starts up successfully the first time in RDB mode, it may then be switched to SDB, redeployed, and restarted. Upon restart, the data in the RDB store will be copied to the SDB store.

This copying process can take a number of hours to complete if the installation contains a large amount of RDF data (roughly a million triples or more). See section Set Up SDB Store in the Background (Optional) for instructions on how to run this lengthy conversion process in the background while an RDB system is operating. Doing this will reduce the time necessary to start VIVO the first time it is run in SDB mode.

The Upgrade Process

1.Download the new distribution file and unpack it into a new source directory.

2.Create deploy.properties, using the same values as in your previous installation and set values for the new variables. The following table shows the default properties for deploy.properties with new V1.2 properties in blue.

Example Value

Default namespace: VIVO installations make their RDF resources available for harvest using linked data. Requests for RDF resource URIs redirect to HTML or RDF representations as specified by the client. To make this possible, VIVO's default namespace must have a certain structure and begin with the public web address of the VIVO installation. For example, if the web address of a VIVO installation is "
`http://vivo.example.edu/`
" the default namespace must be set to "
`http://vivo.example.edu/individual/`
" in order to support linked data. Similarly, if VIVO is installed at "
`http://www.example.edu/vivo`
" the default namespace must be set to "
`http://www.example.edu/vivo/individual/`
"

- **The namespace must end with "individual/" (including the trailing slash).**|

Vitro.defaultNamespace	* `http://vivo.mydomain.edu/individual/` *
Directory where Vitro code is located. In most deployments, this is set to ./vitro-core (It is not uncommon for this setting to point elsewhere in development environments).	
vitro.core.dir	**./vitro-core**
Directory where tomcat is installed.	
tomcat.home	**/usr/local/tomcat**
Name of your VIVO application.	
webapp.name	**vivo**
Directory where uploaded files will be stored. Be sure this directory exists and is writable by the user who the Tomcat service is running as.	
upload.directory	**/usr/local/vivo/data/uploads**
Directory where the Lucene search index will be built. Be sure this directory exists and is writable by the user who the Tomcat service is running as.	
LuceneSetup.indexDir	**/usr/local/vivo/data/luceneIndex**
Specify an SMTP host that the form will use for sending e-mail (Optional). If this is left blank, the contact form will be hidden and disabled.	
Vitro.smtpHost	**smtp.servername.edu**
Specify the JDBC URL of your database. Change the end of the URL to reflect your database name (if it is not "vivo").	
VitroConnection.DataSource.url	**jdbc:mysql://localhost/vivo**
Change the username to match the authorized user you created in MySQL.	
VitroConnection.DataSource.username	**username**
Change the password to match the password you created in MySQL.	

*VitroConnection.DataSource.password	**password**	-*

- Specify the Jena triple store technology to use. SDB is Jena's SPARQL database; this setting allows RDF data to scale beyond the limits of the JVM heap. Set to RDB to use the older Jena RDB store with in-memory caching.*
- -*

■ VitroConnection.DataSource.tripleStoreType*	**SDB**	-*

- Specify the maximum number of active connections in the database connection pool to support the anticipated number of concurrent page requests. It is not necessary to adjust this value when using the RDB configuration.*
- -*

■ VitroConnection.DataSource.pool.maxActive*	**40**	-*

- Specify the maximum number of database connections that will be allowed to remain idle in the connection pool. Default is 25% of the maximum number of active connections.*
- -*

■ VitroConnection.DataSource.pool.maxIdle*	**10**	-*

- Change the dbtype setting to use a database other than MySQL. Otherwise, leave this value unchanged. Possible values are DB2, derby, HSQLDB, H2, MySQL, Oracle, PostgreSQL, and SQLServer. Refer to

```
http://openjena.org/wiki/SDB
/Databases_Supported
```

  for additional information.*
- -*

■ VitroConnection.DataSource.dbtype*	**MySQL**	-*

- Specify a driver class name to use a database other than MySQL. Otherwise, leave this value unchanged. This JAR file for this driver must be added to the the webapp/lib directory within the vitro.core.dir specified above.*
- -*

■ VitroConnection.DataSource.driver*	**com.mysql.jdbc.Driver**	-*

- Change the validation query used to test database connections only if necessary to use a database other than MySQL. Otherwise, leave this value unchanged.*
- -*

■ VitroConnection.DataSource.validationQuery*	**SELECT 1**	-*

- Specify the name of your first admin user for the VIVO application. This user will have an initial temporary password of 'defaultAdmin'. You will be prompted to create a new password on first login.*
- -*

■ initialAdminUser*	**defaultAdmin**	-*

- The URI of a property that can be used to associate an Individual with a user account. When a user logs in with a name that matches the value of this property, the user will be authorized to edit that Individual. For example, to use the netID at Cornell University as the property:

**_seflEditing.idMatchingProperty =

```
http://vivo.cornell.edu/ns/hr/0.9/hr.owl#netId
```

_*

| -

○ selfEditing.idMatchingProperty*	* http://vivo.mydomain.edu/ns#networkId *	-*

- The temporal graph visualization can require extensive machine resources. This can have a particularly noticeable impact on memory usage if*

*:**VIVO is configured to use Jena SDB,**
*:**The organization tree is deep,**
*:**The number of grants and publications is large.**
The VIVO developers are working to make this visualization more efficient. In the meantime, VIVO release 1.2 guards against this impact by disabling the temporal graph visualization unless the "visualization.temporal" flag is set to "enabled". To enable it, uncomment the line for this setting.

| -

○ visualization.temporal*	**enabled**	-*

- The temporal graph visualization is used to compare different organizations/people within an organization on parameters like number of publications or grants. By default, the app will attempt to make its best guess at the top level organization in your instance. If you're unhappy with this selection, uncomment out the property below and set it to the URI of the organization individual you want to identify as the top level organization. It will be used as the default whenever the temporal graph visualization is rendered without being passed an explicit org. For example, to use "Ponce School of Medicine" as the top organization:

**_visualization.topLevelOrg =

```
http://vivo.psm.edu/individual/n2862
```

_*

| -

○ visualization.topLevelOrg*	* http://vivo-trunk.indiana.edu/individual/topLevelOrgURI *	*

3.Apply any previous changes you have made to the new source directory
**
Style="background-color:Lavender; width: 90%"
|-
*|*Special notes regarding source files

- This process assumes any changes made to the application were made in the source directory and deployed, and were not made directly within the Tomcat webapps directory.
- In many cases, simply copying the modified files from your original source directory will not work since the files on which they are baed have changed. It will be necessary to inspect the new source files and add any changes to them at that time.
- NIH-funded VIVO implementations will need to apply the Google Analytics Tracking Code (GATC) to googleAnalytics.ftl in the theme:

[new_source_directory]/themes/[theme_dir]/templates/googleAnalytics.ftl

A sample googleAnalytics.ftl is included in the built-in theme. This file serves only as an example, and you must replace the tracking code shown with your institution's own tracking cod. For additional information about the GATC for the NIH-funded VIVO implementation sites and copy of your institution's tracking code, see the VIVO Google Analytics wiki page.

- If you had used the vivo/contrib/FLShibboleth code in your previous release, you should stop using it. Consult install.html or VIVO Release 1 v1.2 Installation Guide on "Using an External Authentication System with VIVO".

4. If you had modified web.xml to configure the Pellet Reasoner (as described in the installtion instructions), repeat that modification.

5. Stop Apache Tomcat and run ant by typing: ant all

6. Start Apache Tomcat and log in to VIVO.

Ontology Changes

Verify Ontology upgrade process

After Apache Tomcat is started, these files should be reviewed to verify that the automated upgrade process was executed successfully. The ontology alignment process will create the following files in the Tomcat webapps/vivo/WEB-INF directory:

ontologies/update/logs/knowledgeBaseUpdate.log
A log of a summary of updates that were made to the knowledge base and notes about some recommended manual reviews. This file should end with "Finished knowledge base migration". If this file contains any warnings they should be reviewed with your implementation team representative to see whether any corrective action needs to be taken.

ontologies/update/logs/knowledgeBaseUpdate.error.log
A log of errors that were encountered during the upgrade process. This file should be empty if the upgrade was successful.

ontologies/update/changedData/removedData.n3
An N3 file containing al the statements that were removed from the knowledge base.

ontologies/update/changedData/addedData.n3
An N3 file containing all the statements that were added to the knowledge base.

Ontology knowledge base manual review

Changes to the VIVO core ontology may require corresponding modifications of the knowledge base instance data and local ontology extensions.

When Apache Tomcat starts up following the upgrade, it will initiate a process to examine the knowledge base and apply necessary changes. Not all of the modifications that may be required can be automated, so manual review of the knowledge base is recommended after the automated upgrade process. The automated process will make only the following types of changes:

Class or Property renaming
All references to the class (in the subject or object position) will be updated to the new name. References to the property will be updated to the new name.

Class or Property deletion
All individuals in a deleted class will be removed.
All statements using a deleted property will be changed to use the nearest available superproperty. If there is no available superproperty then the statement will be deleted from the knowledge base. Note that all removed and added data is recorded in the files in the changedData directory.

Property addition
If a newly added property is the inverse of a previously existing property, the inverse of any statements using the pre-existing property will be asserted.

Annotation property default values
If a site has modified the value of a vitro annotation (such as displayRankAnnot or displayLimitAnnot) so that it is no longer using the default, then that setting will be left unchanged.
If a site is using the default value of a vitro annotation, and the default has been changed in the new version of the ontology, then the new default value will be propagated to the knowledge base.

File Storage System Upgrade

Changes to the File Storage System

Each uploaded file exists as an individual in VIVO. When the browser requests an uploaded file from VIVO, the data model is queried to find out where the file is actually stored, so it can be downloaded to the browser.

In VIVO 1.2 this storage location, known as the "Alias URL" for the uploaded file, is stored in the file individual. That way, pages that contain many files can be displayed much more quickly.

When Apache Tomcat starts up after the upgrade, it will initiate a process to calculate the "Alias URL" for each existing file and store it in the data model for fast access.

Verify File Storage System upgrade process

The File Storage upgrade process will create a log file in the VIVO upload directory. You should review this file to ensure that this upgrade worked properly.

upgrade/FileStorageAliasAdder-log.2011-00-00T00-00-00.txt
A log of the upgrade process. The actual filename includes a timestamp that tells when the upgrade executed. This file should end withFinished adding alias URLs to FileByteStreams. If this file contains any warnings they should be reviewed with your implementation team representative to see whether any corrective action needs to be taken.

Theme Changes

Introducing a New Default Theme

VIVO 1.2 includes a new default theme called wilma (located in /vivo/themes/wilma) which fully supports all 1.2 features. For details on how to create your own theme using wilma as a starting point, please review the Site Administrator's Guide.

The vivo-basic theme has been deprecated with the 1.2 release and is not recommended for production instances.

Since vivo-basic was the default theme for all previous releases, it is included as part of VIVO 1.2 to help with the transition of upgrading existing installations to the latest code, but all vivo-basic development has ceased and it will not be distributed in future releases.

Please note that vivo-basic does not support all of the new 1.2 features. Most notably, in choosing to use vivo-basic you will be missing out on the following:

*new primary menu for site navigation (replaces tabs)
*home page with class group browse and visual graph
*menu pages with class group and individual browse |

Templates

The 1.2 release continues the transition from JavaServer Pages (.jsp) to Freemarker templates (.ftl) for generating web pages. While there are still JSP files in action behind the scenes, as of 1.2 all theme templates are of the Freemarker variety and are located in the "templates" directory within a theme.

If you did not create a custom theme for your site previously, but used the vivo-basic theme in its original directory, you need not take any action in order to remain using the vivo-basic theme in 1.2.

If you did make changes to the vivo-basic theme, you will need to reapply those changes. We recommend you apply these changes to the wilma theme.

Please note: The vivo-basic theme has been deprecated and is not recommended for production instances.

For details on the new structure of themes in 1.2 and further information regarding the development of your own custom theme, please review the Site Administrator's Guide. This document will focus on updating an existing pre 1.2 theme.

Set UP SDB Store in the Background (Optional)

If your VIVO installation is running in RDB mode, and you'd like to convert to SDB, you can start the conversion process in the background while the RDB system is running. This will reduce the delay in initial startup after the application is redeployed with deploy.properties set for SDB. Note that it is important not to edit any data anywhere in the application while this background conversion is running.

To start the SDB conversion, log in as a system administrator and request /sdbsetup (For example, if your VIVO is installed at vivo.myuniversity.edu/ you would type

```
http://vivo.myuniversity.edu/sdbsetup
```

into your browser).

Click the button that appears on this page.

During the course of the SDB setup, which may take several hours on a large database, subsequent requests to /sdbsetup will display a message that the operation is still in progress. When a request for this page shows a message that the SDB setup has completed successfully, shut down Tomcat, set deploy.properties to SDB mode, redeploy, and restart Tomcat. VIVO will now be running from the SDB store.