VIVO 1.2 Installation Guide

go to Install Guides for all versions

VIVO Release 1 V1.2 Installation Guide

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.

Installation process for V1.2

This document is a summary of the VIVO installation process. This and other documentation can be found on the support page at VIVOweb.org
*These instructions assume that you are performing a clean install, including emptying an existing database and removing a previous installation from the Tomcat webapps directory. Product functionality may not be as expected if you install over an existing installation of an earlier version.
*If you are going to upgrade an existing service, please consult the upgrade.txt in this directory.

VIVO Developers: If you are working on the VIVO source code from Subversion, the instructions are slightly different. Please consult developers.txt in this directory.

Steps to Installation

Install required software

Before installing VIVO, make sure that the following software is installed on the desired machine:
*Java (SE) 1.6 or higher, http://java.sun.com (Not OpenJDK)
*Apache Tomcat 6.x or higher, http://tomcat.apache.org
*Apache Ant 1.7 or higher, http://ant.apache.org
MySQL 5.1 or higher, http://www.mysql.com

Be sure to set up the environment variables for JAVA_HOME and ANT_HOME and add the executables to your path per your operating system and installation directions from the software support websites.

- 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.

Create an empty MySQL Database

Decide on a database name, username, and password. Log into your MySQL server and create a new database in MySQL that uses UTF-8 encoding. You will need these values for Step IV when you configure the deployment properties. At the MySQL command line you can create the database and user with these commands substituting your values for dbname, username, and password. Most of the time, the hostname will equal localhost.

CREATE DATABASE dbname CHARACTER SET utf8;

Grant access to a database user. For example:

GRANT ALL ON dbname.* TO 'username'@'hostname' IDENTIFIED BY 'password';

Keep track of the database name, username, and password for Step IV.

Download the VIVO Application Source

Download the VIVO application source as either rel-1.2.zip or rel-1.2.gz file and unpack it on your web server:

http://vivoweb.org/download

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 trade-off 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.

Specify deployment properties

At the top level of the unpacked distribution, copy the file example.deploy.properties to a file named simply deploy.properties. This file sets several properties used in compilation and deployment.

Windows: For those installing on Windows operating system, include the windows drive and use the forward slash "/" and not the back slash "\" in the directory locations, e.g. c:/tomcat.

External authentication: If you want to use an external authentication system like Shibboleth or CUWebAuth, you will need to set two additional properties in this file. See the section below entitled Using an External Authentication System with VIVO.

Example Value	
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 connection in the database connection pool to support the anticipated number of concurrent page requests. It is not necessary to adjust this value when using 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 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 sued 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.	

*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 organization/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 belowe 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` *

Compile and deploy

At the command line, from the top level of the unpacked distribution directory, type:

ant all

to build VIVO and deploy to Tomcat's webapps directory.

Set Tomcat JVM parameters and security limits

Currently, VIVO copies the contents of your RDF database into memory in order to serve Web requests quickly (the in-memory copy and the underlying database are kept in sync as edits are performed).

VIVO will require more memory than that allocated to Tomcat by default. With most installations of Tomcat, the "setenv.sh" or "setenv.bat" file in Tomcat's bin directory is a convenient place to set the memory parameters.
For example:

br>
export CATALINA_OPTS="-Xms2048m -Xmx1024m -XX:MaxPermSize=128m"

This sets Tomcat to allocate an initial heap of 2048 megabytes, a maximum heap of 1024 megabytes, and a PermGen space of 128 megs. 1024 megabytes is a minimum practical heap size for production installations storing data for large academic institutions, and additional heap space is preferable. For testing with small sets of data, 256m to 512m should be sufficient.

If an OutOfMemoryError is encountered during VIVO execution, it can be remedied by increasing the heap parameters and restarting Tomcat.

apache	hard	nproc	400
tomcat6	hard	nproc	1500

Start Tomcat

Most Tomcat installations can be started by running startup.sh or startup.bat in Tomcat's bin directory. Point your browser to "*

```
http://localhost:8080/vivo/
```

*" to test the application. If Tomcat does not start up, or the VIVO application is not visible, check the catalina.out file in Tomcat's logs directory.

Log in and add RDF data

If the startup was successful, you will see a welcome message informing you that you have successfully installed VIVO. Click the "Log in" link near the upper right corner. Log in with the initialAdminUser username you set up in Step IV. The initial password for the initialAdminUser account is "defaultAdmin" (without the quotes). On first login, you will be prompted to select a new password and verify it a second time.

After verifying your new password, you will be presented with a menu of editing options. Here you can create OWL classes, object properties, data properties, and configure the display of data. Currently, any classes you wish to make visible on your website must be part of a class group, and there are a number of visibility and display options available for each ontology entity. VIVO comes with a core VIVO ontology, but you may also upload other ontologies from an RDF file.

Under the "Advanced Data Tools" click "Add/Remove RDF Data." Note that Vitro currently works best with OWL-DL ontologies and has only limited support for pure RDF data. You can enter a URL pointing to the RDF data you wish to load or upload from a file on your local machine. Ensure that the "add RDF" radio button is selected. You will also likely want to check "create classgroups automatically."

Clicking the "Index" tab in the navigation bar at the top right of the page will show a simple index of the knowledge base.

See more documentation for configuring VIVO, ingesting data, and manually adding data at http://vivoweb.org/support.

Set the Contact Email Address (if using "Contact Us" form)

If you have configured your application to use the "Contact Us" feature in Step IV (Vitro.smtpHost), you will also need to add an email address to the VIVO application. This is the email to which the contact form will submit. It can be a list server or an individual's email address.

Log in as a system administrator. Navigate to the "Site Admin" table of contents (link in the right side of the header). Go to "Site Information" (under "Site Configuration"). In the "Site Information Editing Form," enter a functional email address in the field "Contact Email Address" and submit the change.

If you set the Vitro.smtpHost in Step IV and do NOT provide an email address in this step, your users will receive a java error in the interface.

Set up Apache Tomcat Connector

It is recommended that a Tomcat Connector such as mod_jk be used to ensure that the site address does not include the port number (e.g. 8080) and an additional reference to the Tomcat context name (e.g. /vivo).

This will make VIVO available at "

```
http://example.com
```

" instead of "

```
http://example.com:8080/vivo
```

"

Using the mod_jk connector allows for communication between Tomcat and the primary web server. The Quick Start How To on the Apache site describes the minimum server configurations for several popular web servers.

After setting up the mod_jk connector above, you will need to modify the Tomcat's server.xml (located in [tomcat root]/conf/) to respond to requests from Apache via the connector. Look for the <connector> directive and add the following properties:

connectionTimeout="20000" maxThreads="320" keepAliveTimeout="20000"

Note: the value for maxThreads (320) is equal to the value for MaxClients in the apache's httpd.conf file.

Locate the <Host name="localhost"...> directive and update as follows:

```
<Host name="localhost" appBase="webapps"
      DeployOnStartup="false"
      unpackWARs="true" autoDeploy="false"
      xmlValidation="false" xmlNamespaceAware="false">

  <Alias>example.com</Alias>
  <Context path=""
        docBase="/usr/local/tomcat/webapps/vivo"
        reloadable="true"
        cookies="true" >
    <Manager pathname="" />
    <Environment type="java.lang.String" override="false"
          name="path.configuration"
          value="deploy.properties"
    />
  </Context>
  ...
```

Configure Pellet Reasoner

This optional configuration step is only applicable to VIVO installations running in RDB mode (See section Choose Triple Store for details). VIVO uses the Pellet engine to perform reasoning, which runs in the background at start-up and also when the knowledge base is edited. VIVO continues serving pages while the reasoner continues working; when the reasoner finishes, the new inferences appear. Inferred statements are cached in a database graph so that they are available immediately when VIVO is restarted.

By default, Pellet is fed only an incomplete view of your ontology and only certain inferences are materialized. These include rdf:type, rdfs:subClassOf, owl:equivalentClass, and owl:disjointWith. This mode is typically suitable for ontologies with a lot of instance data. If you would like to keep the default mode, skip to the next step.

To enable "complete" OWL inference (materialize all significant entailed statements), open "vitro-core/webapp/config/web.xml" and search for PelletReasonerSetup.

Then change the name of the listener class to PelletReasonerSetupComplete. Because "complete" reasoning can be very resource intensive, there is also an option to materialize nearly all inferences except owl:sameAs and owl:differentFrom.

This is enabled by specifying PelletReasonerSetupPseudocomplete. For ontologies with large numbers of individuals, this mode can offer enormous performance improvements over the "complete" mode.

Finally, a class called PelletReasonerSetupPseudocompleteIgnoreDataproperties is provided to improve performance on ontologies with large literals where data property entailment's are not needed.

Using an External Authentication System with VIVO

VIVO can be configured to work with an external authentication system like Shibboleth or CUWebAuth.

VIVO must be accessible only through an Apache HTTP server. The Apache server will be configured to invoke the external authentication system. When the user completes the authentication, the Apache server will pass a network ID to VIVO, to identify the user.

If VIVO has an account for that user, the user will be logged in with the privileges of that account. In the absence of an account, VIVO will try to find a page associated with the user. If such a page is found, the user can log in to edit his own profile information.

Configuring the Apache server

Your institution will provide you with instructions for setting up the external authentication system. The Apache server must be configured to secure a page in VIVO. When a user reaches this secured page, the Apache server will invoke the external authentication system.

For VIVO, this secured page is named: /loginExternalAuthReturn

When your instructions call for the location of the secured page, this is the value you should use.

Configuring VIVO

To enable external authentication, VIVO requires three values in the deploy.properties file.

***The name of the HTTP header that will hold the external user's network ID.**

When a user completes the authentication process, the Apache server will put the user's network ID into one of the headers of the HTTP request. The instructions from your institution should tell you which header is used for this purpose.

You need to tell VIVO the name of that HTTP header. Insert a line like this in the deploy.properties file:

externalAuth.netIdHeaderName = [the header name]

For example:

externalAuth.netIdHeaderName = remote_userID

*__Associating a User with a profile page.__

If VIVO has an account for the user, the user will be given the privileges assigned to that account.

In addition, VIVO will try to associate the user with a profile page, so the user may edit his own profile data. VIVO will search the data model for a person with a property that matches the User's network ID. You need to tell VIVO what property should be used for matching. Insert a line like this in the deploy.properties file:

selfEditing.idMatchingProperty = [the URI of the property]

For example:

selfEditing.idMatchingProperty =

```
http://vivo.mydomain.edu/ns#networkId
```

Was the installation successful

If you have completed the previous steps, you have good indications that the installation was successful.
*Step VII showed that Tomcat recognized the webapp, and that the webapp was able to present the initial page.
*Step VIII verified that you can log in to the administrator account.

Here is a simple test to see whether the ontology files were loaded:
*Click on the "Index" link on the upper right, below the logo. You should see a "locations" section, with links for "Country" and "Geographic Location." The index is built in a background thread, so on your first login, you may see an empty index instead. Refresh the page periodically to see whether the index will be populated. This may take some time: with VIVO installed on a modest laptop computer, loading the ontology files and building the index took more than 5 minutes from the time that Tomcat was started.
*Click on the "Country" link. You should see an alphabetical list of the countries of the world.

Here is a test to see whether your system is configured to serve linked data:
*Point your browser to the home page of your website, and click the "Log in" link near the upper right corner. Log in with the initialAdminUser username you set up in Step IV. If this is your first time logging in, you will be prompted to change the password.
*After you have successfully logged in, click "site admin" in the upper right corner. In the drop down under "Data Input" select "Faculty Member(core)" and click the "Add individual of this class" button.
*Enter the name "test individual" under the field "Individual Name," scroll to the bottom, and click "Create New Record." You will be taken to the "Individual Control Panel." Make note of the value of the field "URI" - it will be used in the next step.
*Open a new web browser or browser tab to the page http://marbles.sourceforge.net/. In the pink box on that page enter the URI of the individual you created in the previous step and click "open."
*In the resulting page search for the URI of the "test individual." You should find it towards the bottom of the page next to a red dot followed by "redirect (303)." This indicates that you are successfully serving linked RDF data. If the URI of the "test individual" is followed by "failed (400)" you are not successfully serving linked data.

Finally, test the search index

Type the word "Australia" into the search box, and click on the Search button.You should see a page of results, with links to countries that border Australia, individuals that include Australia, and to Australia itself. To trigger the search index, you can log in as a site administrator and go to "

```
http://your-vivo-url/SearchIndex
```

*".