VIVO 1.6.x Documentation

Upgrade instructions for VIVO release 1.6.2

Author: Jim Blake

Version: 1

Date: Aug 23, 2019 9:16 PM

Table of Contents

Upgrading from Release 1 V1.5 to Release 1 V1.6

This document contains instructions on how to upgrade your installation of VIVO from Version 1.5 (or 1.5.1, or 1.5.2) to Version 1.6. 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 Installing VIVO release 1.6, found on vivoweb.org, or the VIVO Installation Instructions.pdf 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.

For a description of the release contents see the Release announcement for V1.6.

1 Before Performing the Upgrade

Create backups of:

- The VIVO distribution directory (which contains the source for VIVO 1.5 or VIVO 1.5.1)
- The VIVO home directory (pointed to by your deploy.properties file)
- The webapps directory in Tomcat
- MySQL database (most people use mysqldump to create the backup)

If you have used temporary models in the database to stage ingested data, you will want to clear out any unneeded models that remain listed on the `Manage Jena Models` page (under `Ingest tools`). This step is especially important if these temporary models contain blank nodes, as this may cause unwanted or duplicate data to appear following the upgrade.The upgrade process is similar to the initial install process with the following exceptions:

- You do not need to reinstall MySQL or recreate the MySQL database. Please backup your MySQL database as noted above.
- The root account will keep the password that was previously set on it. It will not return to the default password.
- When Apache Tomcat starts up after the upgrade, an automated process will modify the knowledge base to align the data with any ontology updates made for the new release. See the section on the Knowledge Base Migration below for more information.

2 Noteworthy Changes

2.1 VIVO becomes more portable

- The VIVO build script now includes a `distribute` target that will produce a file called `distribution.tar.gz`. This compressed archive contains these files:
    1. `vivo.war` -- a WAR file for the main VIVO application.
    2. `vivosolr.war` -- a WAR file for the Solr application.
    3. `solrhome.tar` -- a Solr home directory that is configured for use with VIVO.

    These files can be used with Tomcat, or with any container that supports the Java Servlet 2.4 Specification.To permit this portability, the `deploy.properties` file has been split in two. `build.properties` contains only the properties that are required for building VIVO. `runtime.properties`, which must be created in the Vitro home directory, contains the properties that VIVO uses while running.If you are building to `deploy` to Tomcat (as with previous releases), then `build.properties` must contain these properties:
    - `vitro.core.dir`
    - `webapp.name`
    - `tomcat.home`
    - `vitro.home`-- *note that this was `vitro.home.directory` in previous releases*

    If you are building to `distribute`, the `build.properties` file requires only these properties:
    - `vitro.core.dir`
    - `webapp.name`

2.2 Solr is no longer secured

In previous releases, Solr was deployed to Tomcat with a `RemoteAddrValve` that would only permit access from certain IP addresses. Acceptable IP addresses were those which matched the regular expression pattern in the `vitro.local.solr.ipaddress.mask` property.This has been removed because:

- It caused repeated problems for sites who were experimenting with VIVO.
- It was not standards-based, but specific to Tomcat.

- It was redundant. Production instances of VIVO are usually hidden behind a firewall and accessed through an Apache Http server.

Sites that need to secure Solr are now left to their own devices.

2.3 Log4J properties file renamed

In previous releases, the properties file for the VIVO logging system was called `default.log4j.properties`. In release 1.6, this file has been renamed to `log4j.properties`. This is so the developers and implementers will know where to look for the file.Note that `debug.log4j.properties`, if present, will still override the default.

2.4 Property groups now displayed in a tab format, including a "View All" tab

With release 1.6, the property group menu bar that was used on profile pages has been replaced by java script enabled tabs. When clicked, each property group tab will display the properties within that group while the contents of the previously displayed group will be hidden. The array of tabs also includes a "View All" tab that, when clicked, displays the contents of all the property groups.

2.5 Class-specific SPARQL Query Data Getters

The VIVO software now supports the development of SPARQL query data getters that can be associated with specific ontological classes. These data getters, in turn, can be accessed within Freemarker templates to provide richer content on VIVO profile pages. For example, the profile page for an academic department lists only the names of the faculty within that department and their titles, but with a SPARQL query data getter it is now possible to extend the faculty information to display all of the faculty members' research areas. Refer to this wiki page for details on how to use class-specific SPARQL query data getters: `https://wiki.duraspace.org/display/VIVO/Enriching+VIVO+Content+Using+SPARQL+Query+Data+Getters`.

2.6 The foaf:Person template has been re-located

The template individual--foaf-person.ftl has been moved to the "templates" subdirectory in the wilma theme directory (vivo/themes/wilma/templates). If your installation has a customized version of individual--foaf-person.ftl, ensure that it is located in the templates subdirectory in your installation's theme directory or, if your installation does not have it's own theme directory, in the `themes/wilma/templates subdirectory`.

2.7 Multiple foaf:Person Profile Pages

VIVO now supports multiple profile pages for foaf:Persons. This feature, which is optional so installations can continue to use just the individual--foaf-person.ftl template, currently consists of two profile page types: a standard view, which is a redesigned version of the foaf:Person template in previous releases; and a quick view, which emphasizes the individual's own web page presence while providing summary VIVO information, such as current positions and research areas. The profile quick view requires the use of a web service that captures images of web pages. **This web service is not included with the VIVO software.** An installation will either have to develop their own service or use a third-party service, usually for a small fee depending on the number of images served. (Examples of these services include WebShotsPro, Thumbalizr and Websnapr.) For more information on how to implement multiple profile page views, refer to this wiki page: `https://wiki.duraspace.org/display/VIVO/Multiple+foaf%3APerson+Profile+Pages`.

2.8 Home Page Re-design

For Release 1.6 the VIVO Home Page has been redesigned. The Search field beneath the "welcome" text now allows the user to limit the results of a search to a specific class group, such as people, organizations, etc. In addition, the browse-by-class-group display has been removed from the Home Page and replaced by multiple features, which include: a list of four randomly selected faculty members, including their titles and thumbnail images; a display of statistical data about the VIVO installation, such as the number of people, activities and organizations; and, optionally a global map showing researchers' areas of geographic focus.

2.9 Auto-loaded RDF files move to the Home directory

The RDF files that initialize the data model have moved, in both the distribution and the runtime locations. In the distribution, they have been collected in one place, and reorganized to use a more consistent naming scheme. During the build process, they are copied to a location within the VIVO home directory, instead of residing in the webapp itself.If you have modified these RDF files, or added files of your own, you must adjust to the new locations accordingly.

Old locations of RDF files under [Vitro]/webapp/web or [VIVO] /productMods	**New locations of RDF files under [Vitro]/webapp/rdf or [VIVO]/rdf**	**Comments**
WEB-INF/ontologies/app/	rdf/display/firsttime/	
WEB-INF/ontologies/app /loadedAtStartup/	rdf/display/everytime/	
WEB-INF/ontologies/app/menuload /displayTBOX.n3	rdf/displayTbox/everytime/	Was one file, now a directory
/WEB-INF/ontologies/app/menuload /displayDisplay.n3	rdf/displayDisplay/everytime/	Was one file, now a directory
WEB-INF/ontologies/user /applicationMetadata/ WEB-INF/init-data/	rdf/applicationMetadata /firsttime/	Merged directories
WEB-INF/ontologies/user/abox/	rdf/abox/firsttime/	
WEB-INF/filegraph/abox/	rdf/abox/filegraph/	
WEB-INF/ontologies/user/tbox/	rdf/tbox/firsttime/	
WEB-INF/filegraph/tbox/	rdf/tbox/filegraph/	

If you are using a three-tier build process, you will need to add two lines to the build script to accomodate the RDF files, and the language support (see below) So this:

```
<patternset id="appbase.patterns">
  <include name="src/**/*" />
  <include name="lib/**/*" />
  <include name="test/**/*" />
  <include name="themes/**/*" />
  <include name="config/*.properties" />
  <include name="config/*.txt" />
  <include name="config/jarlist/*.txt" />
  <include name="config/solr/*" />
  <include name="context.xml" />
</patternset>
```

becomes this:

```
<patternset id="appbase.patterns">
  <include name="src/**/*" />
  <include name="lib/**/*" />
  <include name="rdf/**/*" />
  <include name="languages/**/*" />
  <include name="test/**/*" />
  <include name="themes/**/*" />
  <include name="config/*.properties" />
  <include name="config/*.txt" />
  <include name="config/jarlist/*.txt" />
  <include name="config/solr/*" />
  <include name="context.xml" />
</patternset>
```

2.10 Support for additional languages

VIVO 1.6 includes limited support for other languages, in addition to American English. This limited support is described as *read-only* support on *public-facing* pages.*Read-only* means that there is no provision for editing multi-language data or displays. Property values, ontology labels, etc. must all be provided in RDF files and ingested or otherwise inserted into the data model. The Page Management user interface does not support maintaining pages in multiple languages.*Public-facing* means that most of the pages used for site adminstration are only presented in American English.These two pages in the VIVO Wiki describe how to Build VIVO with multiple languages and how to Add a new language to VIVO.

2.11 More compact responses to Linked data requests

In VIVO 1.6, the response to requests for linked data is changed, to be smaller and faster. When responding to a request for linked data about an individual, VIVO 1.6 returns:

- Data properties of the individual
- Object relationships to and from the individual
- The RDF types and RDFS labels for any object that directly relates to the individual

This data is filtered by the usual VIVO privacy policies, so properties such as salary or employee ID number may not be revealed unless the requester has been properly authenticated.VIVO releases prior to VIVO 1.6 returned a more complex set of statements, referred to as "extended linked data":

- Data properties of the individual
- Object relationships from the individual
- All properties of the context nodes (positions, roles, etc.) that are associated with the individual.
- Labels of objects that are joined to the individual through context nodes.
- Full details of time intervals that are attached to context nodes: start, end, precision.

As above, this data was filtered by the VIVO privacy policies. Although these additional items were included, extended linked data was based only on relationships from the individual. Relationships to the individual were not included.Extended linked data was costly to produce, in terms of resources, because it required a recursive search of the data model. Extended linked data typically contained 50% more information than its non-extended equivalent, and took more than 10 times as long to produce.VIVO release 1.6 can be configured to produce extended linked data like previous releases. However, extended linked data will not be supported in future releases.

2.12 Changes to default types for Google Refine

The list of default types for Google Refine has changed, to accommodate changes in the ontology. If you are using Google Refine, you may need to change your runtime properties accordingly. The new defaults appear in `example.runtime.properties`.

2.13 Special runtime settings for developers

This release includes several settings to help developers by instrumenting the Freemarker templates and the SPARQL queries on the RDFService. Check the wiki for details, or look in the home directory in `example.developer.properties`.

2.14 Supported Browsers

For this release, the following browsers are supported.

- Mac:
  - Chrome 30.0.1599.69 and above
  - FireFox 3.6.28, 10.0.12, 24
  - Opera 12.02
  - Safari 5.0.3
- PC:
  - Chrome 25.1364.2 and above
  - FireFox 10.0.12, 24
  - Internet Explorer 8, 9, 10
  - Opera 12.02

3 Upgrade Instructions

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

3.2 Separate your existing deploy.properties file into two files, as described below.

Store the new `build.properties` file in the top level of the VIVO distribution directory. Store the new `runtime.properties` file in your VIVO home directory.

Properties in `build.properties`	**Properties in `runtime.properties`**

`vitro.core.dir`
`vitro.home`
`tomcat.home`
`webapp.name`
	All other properties from `deploy.properties`

Note that `vitro.home` replaces `vitro.home.directory` *Note that `vitro.local.solr.ipaddress.mask` is no longer used.*	

If you prefer, you may start with `example.build.properties` and `example.runtime.properties`, make copies, and edit them to suit your installation. Remember, the `runtime.properties` file goes into your VIVO home directory.The properties below are new to `build.properties`. They are optional, so you need not add them unless you want a value other than the default.

Property Name	**Example Value**
Languages (in addition to American English) that will be built into your VIVO site. The languages must be found in the `languages` directory of the VIVO distribution. See the VIVO Wiki for more information.	
languages.addToBuild	es_MX

The properties below are new to `runtime.properties`. They are optional, so you need not add them, unless you want a value other than the default.

Property Name	Example Value
Tell VIVO to generate HTTP headers on its responses to facilitate caching the profile pages that it creates. This can improve performance, but it can also result in serving stale data. Default is false if not set. For more information, see the VIVO wiki page: Use HTTP caching to improve performance	
http.createCacheHeaders	true
Force VIVO to use a specific language or Locale instead of those specified by the browser. This affects RDF data retrieved from the model, if RDFService.languageFilter is true. This also affects the text of pages that have been modified to support multiple languages.	
languages.forceLocale	en_US
A list of supported languages or Locales that the user may choose to use instead of the one specified by the browser. Selection images must be available in the i18n/images directory of the theme. This affects RDF data retrieved from the model, if RDFService.languageFilter is true. This also affects the text of pages that have been modified to support multiple languages.	
languages.selectableLocales	en, es, fr_FR
On the VIVO home page, display a global map highlighting the geographical focus of foaf:person individuals. The default is `enabled`.	
homePage.geoFocusMaps	enabled
MultiViews for foaf:person **profile pages.** VIVO supports the simultaneous use of a full foaf:Person profile page view and a "quick" page view that emphasizes the individual's own webpage presence. **Implementing this feature requires an installation to develop a web service that captures images of web pages or to use an existing service outside of VIVO, usually for a small fee.** The default is `disabled`.	
MultiViews.profilePageTypes	disabled

Property Name	Example Value
Setting this property causes VIVO 1.6 to produce extended responses to requests for linked data. This provides compatibility with earlier releases. The default is `false`. Extended linked data is costly, in terms of server resource. Typically, extended linke data contains 50% more information than its non-extended equivalent, and takes 10 times as long to produce. **Extended linked data will not be supported in future releases of VIVO.**	
serveExtendedLinkedData	true

Note that the property named `externalAuth.buttonText` is no longer used. You can specify the text of the external login button by adding a property to `all.properties` like this:

```
external_login_text = Log in using BearCat Shibboleth
```

3.3 Apply any previous changes you have made to the new source directory.

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 based 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/googleAnal`

  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 code. For additional information about the GATC for the NIH-funded VIVO implementation sites and a copy of your institution's tracking code, see the VIVO Google Analytics wiki page.

3.4 Apply any previous changes you have made to the RDF initialization files.

See the section on the Auto-loaded RDF files above for more details.

3.5 Run the build script

Stop Apache Tomcat and from your VIVO source directory, run ant by typing: `ant all`

3.6 Start VIVO

Start Apache Tomcat and log into VIVO as the root user when the upgrade is completed. Depending on the size of your database, the migration process may take up to several hours. When it is complete, you will see a message in the catalina.log file that the server has started.

```
INFO: Server startup in XXXXX ms
```

3.7 Rebuild the search index

As root or an administrator, request a rebuild of the Solr search index: Go to the "Site Admin" page and click on "Rebuild Search Index" under the heading "Site Maintenance".

3.8 Review the knowledge base migration logs.

The knowledge base migration process described in the next section will create logs in a subdirectory of the VIVO home directory:(home directory)/upgrade/knowledgeBase/logs /knowledgeBaseUpdate.(timestamp).log A log of a summary of updates that were made to the knowledge base. 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. (home directory)/upgrade /knowledgeBase/logs/knowledgeBaseUpdate.error.(timestamp).log A log of errors that were encountered during the upgrade process. This file should be empty if the upgrade was successful. If any errors are encountered you will need to rerun the knowledge base migration.

3.9 Load the About Page .N3 file (optional).

Release 1.6 provides an "about VIVO" page that is editable through the GUI, using the Page Management functionality. If your installation has a customized version of the About Page and you do not need to have it accessible via Page Management, then skip this step. Otherwise, here are the instructions for loading the About Page .N3 file:

1. Once the VIVO application is running, go to the Site Admin page and click the "Ingest Tools" link under Advanced Data Tools.
2. From the Ingest Menu click the "Manage Jena Tools" link, and then click the "RDB Models" button.
3. Locate the "vitro-kb-displayMetadata" model and click the "load RDF data" button.
4. Click the "Browse" button to upload the file from your computer and select the aboutPage.n3 file located here in the VIVO source: `productMods/WEB-INF/ontologies/app/aboutPage.n3`.
5. Select N3 as the file type from the drop-down list and then click the "Load Data" button.
6. Restart tomcat.

If your installation has a customized version of the About Page, but you would like to make its content editable through the GUI, follow the above steps and then use Page Management to update the fixed HTML content.

4 Knowledge Base Migration

Changes to the VIVO core ontology may require corresponding modifications to the knowledge base instance data and ontology annotations. The first time VIVO starts up following the upgrade, it will initiate a process to examine the knowledge base and apply necessary changes. The knowledge base migration process for release 1.6 will make the following types of changes:

- Instance data changes to align with VIVO-ISF
- Obsolete predicates and types in the instance data will be updated where necesary to correspond to the current properties and classes in version 1.6 of the VIVO-ISF ontology. Note that VIVO 1.6 continues to make use of certain deprecated classes and properties that are not incompatible with VIVO-ISF. Most notable among these are the various subclasses of foaf:Person: these types will be unchanged by the VIVO 1.6 data migration, but may be removed in a future release.
- 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. Note that the annotation settings for certain obsolete VIVO 1.5 properties will be preserved in a configuration file that applies settings to particular object properties based on the types of their subject and object individuals. This file is found in the VIVO home directory: `(home directory)/rdf/display/everytime /PropertyConfig.n3`
- 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.

In addition to the logs described in **step 8** of the previous section, the knowledge base migration process will log copies of all additions and deletions that were made to the knowledge base in the following files in the VIVO home directory:

`(home directory)/upgrade/knowledgeBase/changedData/removedData.(timestamp).n3`

An N3 file containing all the statements that were removed from the knowledge base.

`webapps/vivo/WEB-INF/ontologies/update/changedData/addedData.(timestamp).n3`

An N3 file containing all the statements that were added to the knowledge base.

5 Review the VIVO Terms of Use

VIVO comes with a "Terms of Use" statement linked from the footer. The "Site Name" you assign in the "Site Information" form under the **Site Admin** area will be inserted into the "Terms of Use" statement. If you want to edit the text content more than just the "Site Name", the file can be found here:

`[vivo_source_dir]/vitro-core/webapp/web/templates/freemarker/body/termsOfUse.f`

Be sure to make the changes in your source files and deploy them to your tomcat so you don't lose your changes next time you deploy for another reason.

6 Next Steps

Now that you have VIVO up and running, please refer to the Site Administrator's Guide for information about its operation.