VIVO 1.10.0 Release Testing

The next major release of Vitro / VIVO contains an upgrade of Jena - from 2.10.1, to Jena 3.x (3.1.1 at time of writing). Whilst development is still progressing, in order to maximise amount of testing of this core upgrade, a beta has been released.

The procedure here describes testing an upgrade from an earlier version of VIVO. If you have questions, please post to vivo-tech@googlegroups.com

What has changed

Jena 3 improves Jena's RDF 1.1 compatibility. Specifically, literal values are always stored internally with datatypes. "Untyped" string literals are the same as the identical value typed as xsd:string. See the following document for more information

> https://jena.apache.org/documentation/migrate_jena2_jena3.html

For VIVO, this means that the two triples:

```
<subj> <pred> "value"
<subj> <pred> "value"^^xsd:string
```

will be treated as the same triple and only stored once in your triple store. As a result, when using the procedure described below to upgrade your triple store, you may find that the number of triples in your triple store after the upgrade is lower than the number before the upgrade.

Any code, tools, parsers, utilities, or queries that expect to differentiate between these two triples will produce results different than produced previously – RDF 1.1 no longer distinguishes between these two triples. In particular, queries that limit results based on the xsd:string datatype will likely produce larger result sets, as previously untyped triples are now typed as xsd:string internally.

Another way of saying this – any triple loaded into VIVO or Vitro that does not have a type will be typed as xsd:string internally.

Exports from VIVO and Vitro will never include the xsd:string datatype. Literal values that do not have explicit types are always assumed to be xsd:string.

As a result, we recommend that input process for VIVO do not include xsd:string datatypes on literals. While they may be correct, and will result in the literal value being typed as xsd:string internally, export processes will not include the xsd:string on output.

In RDF 1.0, a type could not be asserted with a language identifier. In RDF 1.1, a type can be asserted with a language identifier. Untyped input with language identifiers were left as untyped internally in RDF 1.0. In RDF 1.1, untyped input with language identifiers are assumed to have type rdf:langString. Exports from VIVO for triples with language tags will never include the rdf:langString datatype. Literal values with language tags are always assumed to be rdf:langString.

As a result, we recommend that input process for VIVO do not include datatypes on triples with language types. While asserting rdf:langString is correct, and will result in the literal value being typed as rdf:langString internally, export processes will not include the rdf:langString on output.

Code, tools, parsers, utilities based on RDF 1.0 should not be used with Vitro and VIVO 1.10. All code, tools, parsers, and utilities must support RDF 1.1.

On start-up of version 1.10, the triple store is checked to insure that it has been upgraded. If untyped literals are found in the triple store, an error message will appear in the browser and the application will not start. The test applies only to the content store. It is possible that your content store could pass this test, but your configuration triple store remains incompatible with Jena 3 and RDF 1.1. In such a case, your application may become unstable. The procedure below will upgrade both your configuration triple store and your content triple store.

Upgrade procedure

It is required that you reload any SDB and TDB triple stores when upgrading to Jena 3 using the procedure and tools described below.

Warning

VIVO/Vitro uses two triple stores – a configuration triple store typically stored using TDB in <vivo home dir>/tdbModels, and a content triple store typically stored using SDB in MySQL. The procedures described below assume that you are running this standard configuration. If you are not, you will need a custom procedure for upgrading your triple stores.

Step 1: Shutdown Tomcat

Use your local procedure for shutting down Tomcat. Tomcat must be shut down for the upgrade process to proceed.

Step 2: Install VIVO 1.10

Download 1.10 beta from GitHub. Follow the instructions for installing VIVO. Stop prior to starting Tomcat.

Step 3: Export the triple stores

- ***To export successfully, you need to ensure no other programs are accessing your triple stores.***
- ***Your file system must have the space available and be capable of storing files large enough to contain your entire triple store serialisation.***

To export your triples store, use the jena2tools utility provided with VIVO 1.10, in <vivo home dir>/bin, specifying the export command, as shown below.

```
java -jar jena2tools.jar -d <vivo home dir> -e
```

Arguments:

-d - the location of the Vitro/VIVO home directory

-e - run in export mode

On execution, the program will read your configuration files, find your Vitro or VIVO configuration within the vivo/vitro home directory, and get the necessary information to connect to your configuration triple store (usually <vivo home dir>/tdbModels), and your content triple store (usually in SDB). If your triple store(s) are not SDB or TDB backed, then it will simply skip them.

jena2tools will then extract the contents of the available triple stores, and dump them to <vivo home dir>/dumps in TriG format.

Step 4: Inspect the dumps

Check <vivo home dir>/dumps to confirm that the triple stores have been exported. Inpect the dump files to insure they contain the data from your triple stores.

Step 5: Empty your triple stores

Drop all tables in your SDB database as named in your runtime.properties. You may drop your database and recreate it as empty, just as you would for creating a new VIVO install. jena3tools must find an empty database (no tables) as named in your runtime.properties and will recreate your SDB triple store as tables in the named database using the triples produced by jena2tools and stored in <vivo home dir>/dumps/content.trig

Delete all files in <vivo home dir>/tdbModels. Jena3tools will rebuild your configuration tdbModels based on the content created by jena2tools and stored in <vivo home dir>/dumps/configuration.trig

Step 6: Import the triple stores

Having exported your Jena 2 triple stores, you can reload them using jena3tools, also available with VIVO 1.10, specifying the import command.

```
java -jar jena3tools.jar -d <vivo home dir> -i
```

Arguments:

-d - the location of the Vitro/VIVO home directory

-i - run in import mode

On execution, the program will find your Vitro or VIVO configuration within the home directory, as well as the dumps that you have created with jena2tools. It will import them into the SDB and TDB triple stores, based on the configuration of your Vitro/VIVO instance.

jena3tools will be present in <vivo home dir>/bin when you install the 1.10 beta. Alternatively, it can be downloaded from GitHub.

Step 7: Start Tomcat

Using your normal procedure, start Tomcat. Perform your usual start-up tests – login, view pages, conduct searches, examine visualizations, perform queries. Please report your findings to vivo-tech@googlegroups.com

When starting Tomcat, does Vitro / VIVO start as expected?

After upgrading, does the application behave as expected? Can you see everything in VIVO, log in, edit content, use the system admin pages, see visualizations, conduct searches, etc.

Reporting

If you are testing the Vitro / VIVO beta release, please report your findings:

- What version of Vitro / VIVO you upgraded from?
- What storage was used for the content triple store (SDB or TDB)?
- Did it work as expected? If not, what happened?

Please report your findings on the vivo-tech@googlegroups.com mailing list.

Files

https://github.com/vivo-project/VIVO/releases