Additional Considerations

Overview

If your site does not have customizations, and does not use Jena-based applications, these considerations do not apply.

Upgrade Local Java Code Using Jena

If you have local customisations or additional applications that make use of the Jena libraries, you will need to upgrade these to work with Jena 3. Mostly this is simply a case of renaming any packages in imports for Jena classes:

import com.hp.hpl.jena.*

becomes

import org.apache.jena.*

However, some classes have been moved, or removed, and some interfaces have additional methods. So in rare cases you may find that you need to make a few small changes beyond this.

Notes on Other Dependency Changes

To remove the possibility of incompatible classes being loaded, and to remove known vulnerabilities from the code base, most of the Java dependencies in VIVO have been removed, updated or replaced.

For the most part, this will have minimal impact on local customisations.

Some libraries - such as commons-lang3 - have new package names, but mostly compatible classes, and usually just require the imports to be adjusted.

The CSV libary was outdated and unmaintainted, and has been replaced with commons-csv.

There were originally 5 JSON libaries (Jackson, Gson, Glassfish, Sourceforge.net and Json.org parsers). All code is now using Jackson, and the other parsers have been removed.

38 dependencies have been removed from the standard distribution. If you have any customisations that make use of them, that should work, but you will need to add the dependencies to your own projects.

22 dependencies will appear to have been added, but these are mostly from unbundling the owlapi OSGi dependency, and these - along with other conflicting classes - had been packaged as part of the bundle.

while every effort has been made to eliminate known vulnerabilities, the Maven dependency-check plugin still reports 13 vulnerabilities across 8 libraries - for which these are currently the latest releases.

In total, 11 out of an original 113 dependencies are still at the same version as the previous release.

UI Changes

jQuery 1.12.4

Bootstrap based themes require a newer version of jQuery than was shipped with VIVO. In order for all of the functionality aross the UI to work, and to minimise the amount of duplication in the UI, it was necessary to upgrade all of the pages and plugins that used jQuery, so that the same upgraded version works across all of the pages and themes.

This release does require some migration of javascript that makes use of it. There is a script - jquery-migrate.js that helps with this, providing extra backwards compatibility, and logging warnings to the console whenever an old method is used.

All of the existing code that was relying on the migrate script has been updated, so that it is no longer needed.

To aid transition, VIVO still ships with the jquery-migrate.js script, allowing most existing code to work. You should monitor the Javascript console, and apply updates if you see any logged messages in your customisations - future versions of VIVO will remove this migration script in order to upgrade to later versions.

jQuery plugins

In order to support the upgrade to jQuery 1.12.4, and remove any backward compatibility messages from the Javascript console, the following plugins have been upgraded:

jQuery UI

jCrop

qTip

DataTable

In addition, the following plugins have been replaced with equivalents for compatibility and/or licensing issues:

Old	New	Notes
mb.FlipText	Jangle	Used for rotating text on the graphs (e.g. temporal graph)
isotope	wookmark	Used to render the three columns on the index page

D3 v4

This is another major upgrade that has architectural changes to improve modularity and make writing visualizations easier, as well as a selection of new features.

Where D3 was being included by VIVO (e.g. on the profile page, in co-authorship networks, etc.), these now include D3 v4, and the visualizations have been upgraded to use D3 v4.

The only visualization that has NOT been upgraded to D3 v4, is the Capability Map - as a full page visualization, that page is still including it's own D3 v3.

When upgraading, you have a few choices:

1) If you have a full page custom visualization, you can continue to specify whatever libraries and versions that you want to use, although if you are referencing the "shared" D3, you will need to adjust this to include your own D3 that is the version you require.

2) If you are embedding the visualization on a page that may have other visualizations, you should either:

a) Upgrade your custom visualization to use D3 v4.

b) Render your visualization into an iframe, so that you can specify your own javascript dependencies.

By making this change now, we are getting on to a maintained version of D3 (v3 has not had any releases for 18 months), and it prepares us for using D3.express in the future, for more dynamic visualization building.

List View Configurations

To produce complex views of data associated with properties (e.g. a person's publication list), VIVO allows the association of externalized SPARQL queries, and associated Freemarker template links.

The configuration for this are the files called "listViewConfig-*" in the <webapp>/configs/ directory.

Due to the performance of OPTIONAL clauses for some triple stores, in VIVO 1.x the configuration files usually consist of a SELECT query (to retrieve the data for the associated Freemarker template), and one or more CONSTRUCTs (using UNIONs) to create a smaller model that the SELECT query would then be performed against.

This creates additional buffering of an intermediate model, the overhead of performing multiple queries, and makes the configurations harder to write and maintain.

For triple stores with poor OPTIONAL performance, this is a result of the OPTIONAL clause returning large amounts of data that is then joined to the rest of the query.

This can be avoided by making the OPTIONAL clauses contain the restrictions that they will be joined to, in addition to them appearing outside for the join.

As this is not necessary for all triple stores, an element <precise-subquery> has been introduced, so that these additional statements can be filtered out for triple stores where they aren't required.

So,

```
SELECT ?menuItem
       ?linkText
 WHERE {
    ?subject ?property ?menuItem .
    OPTIONAL {
        ?menuItem display:linkText ?linkText .
    }
}
```

can be rewritten as

```
SELECT ?menuItem
       ?linkText
 WHERE {
    ?subject ?property ?menuItem .
    OPTIONAL {
        <precise-subquery>?subject ?property ?menuItem .</precise-subquery>
        ?menuItem display:linkText ?linkText .
    }
}
```

When all OPTIONAL clauses are rewritten like this, the <query-construct> elements can be removed, for approximately 10% performance improvement, and a larger reduction in Java processing overhead. This allows the web server to scale to handle more requests.

If you have customized listViewConfig-*.xml files, you do not need to rewrite them for VIVO 1.10 - they will work unmodified. However, you will have a small performance improvement, and more readable, maintainable queries, if you choose to modify them to take advantage of the new features.

Servlet 3.0 Upgrade

The web.xml that ships with VIVO has been updated to use 3.0 semantics (the required version of Tomcat already supported 3.0).

If you have customisations that introduce new servlets, then you can still add the configuration to web.xml, or you can modify the servlet to use @WebServlet annotations.

If you are replacing a servlet, then you will need to map the existing servlet to an unused url, and map the new servlet by adding configuration for these servlets to web.xml.

Alternatively, if you want to disable a servlet, then you can set the web.xml back to 2.5 spec, and add all of the servlet configuration explicitly to the web.xml.

Java Dependencies

HttpClient

Due to OSGi bundling of some of the core dependencies, VIVO had been shipping three different versions of HttpClient, all of which were incompatible with each other, and sometimes generated errors depending on which code paths got executed first.

VIVO 1.10 brings convergence around HttpClient 4.5.3, removing the problems caused previously. If you have any customisations that depend on HttpClient (or the fluent api - fluent-hc), please ensure that you are using the versions that are already indlueded with VIVO. This may require some minor adjustments to your client code to make it compatible.

OSGi Dependencies

The OWLAPI previously included with VIVO was an OSGi bundle, and included the classes of a number of libraries (such as HttpClient above). This causes classloading conflicts. In VIVO 1.10, we are depending directly on the libraries that were being bundled, rather than the bundle in it's entirety.

In addition, Jena has OSGi dependencies for HttpClient, leading to more duplicate classes. These have been explicitly excluded - see dependencies/pom.xml for how this is done.

JSON Parsers

Four JSON parsers - GSON (com.google.gson), Glassfish (javax.json.Json, Sourceforge.net (net.sf.json) and JSON.org (org.json) have been removed, to simplify the code base, reduce vulnerabilities and improve performance.

All JSON parsing in VIVO is now handled through Jackson.

If you have local customisations that use a different JSON parser, you can add the dependency to your projects, although it is recommended that migrating to Jackson is preferable for long term maintenance.

Replaced Dependencies

The CSV parser has been replaced with commons-csv.

Removed Dependencies

The following dependencies have been removed from VIVO. If you have customisations that require any of them, you will need to add them as dependencies in your own projects.

agrovocws-3.0.jar, asm-3.1.jar, aterm-java-1.8.2.jar, axis-1.3.jar, axis-jaxrpc-1.3.jar, axis-saaj-1.3.jar, bcmail-jdk14-1.38.jar, bcprov-jdk14-1.38.jar, bctsp-jdk14-1.38.jar, c3p0-0.9.2-pre4.jar, cglib-2.2.jarcommons-beanutils-1.7.0.jar, commons-discovery-0.2.jar, cos-05Nov2002.jar, csv-1.0.jar, cxf-xjc-runtime-2.6.2.jar, cxf-xjc-ts-2.6.2.jar, dom4j-1.6.1.jar, ezmorph-1.0.4.jar, gson-2.5.jar, jai_codec-1.1.3.jar, jai_core-1.1.3.jar, JavaEWAH-0.8.6.jar, javax.json-api-1.0.jar, jjtraveler-0.6.jar, jsonld-java-jena-0.2.jar, json-lib-2.2.2-jdk15.jar, lucene-analyzers-common-5.3.1.jar, lucene-core-5.3.1.jar, lucene-memory-5.3.1.jar, lucene-queries-5.3.1.jar, lucene-queryparser-5.3.1.jar, lucene-sandbox-5.3.1.jar, mail-1.4.jar, mchange-commons-java-0.2.2.jar, shared-objects-1.4.9.jar, sparqltag-1.0.jar, stax-api-1.0-2.jar, wsdl4j-1.5.1.jar