**Provisium Architecture**

The API currently supports the following calls:

Submission – 3 inputs (apiKey, provenance text, provenance encoding (turtle, rdf-xml, n3, or n-quads) – returns a URL to where the prov document is hosted online (for sharing and reuse outside of the triple store)

Update – 4 inputs (apiKey, provenance text, provenance encoding (turtle, rdf-xml, n3, or n-quads), and url (returned in previous call to Submission) – the file at url is updated with the new provenance text and the triple store is updated as well

NewKey – 3 inputs (newKey, adminKey, name) – an administrative utility for creating new api keys, set up as a POST call to allow for new key creation from outside of Provisium, “name” is an organizational or person name to associate with newKey

The API configuration variables can be found in config/config.json and are read at startup time.

Description of configuration variables is as follows:

"WebStorageDir" – a web accessible directory into which provenance

documents will be written

"Port" – the port Provisium API should listen on

"URL" – a prefix for all URLs generated by Provisium, e.g

<http://localhost/> or <http://provisium.io/> (be sure to include

the trailing /)

"LogFile" – full path and filename to the log file, e.g.

./logs/serviceslog.txt",

"ApiKeys" – admin password for creating new api keys

"GraphUriBase" – URL prefix to use for triplestore named graphs, e.g

<http://provisium.io/prov/>

"KvStoreAPI" – full path and filename of the key value store to use

for API calls, e.g. ./dataVolume/kvStores/api.db

The API workflow is as follows:

1. Upon successful validation of API key and input parameters the PROV document is written to a subdirectory in WebStorageDir. The subdirectory is the API key. For example, if the API key is 1234 and WebStorageDir is /some/dir/ then the PROV is written to /some/dir/1234. This is intended to make triple store updates easier. Each API key has its own named graph in the triple store. When updates happen, the named graph is deleted, and a new named graph is created will all the files in /some/dir/1234. Otherwise, we’d have to selectively delete just the triples affected by the update, which can get complicated.
2. The API returns a URL to the PROV file that was just written. The PROV documents are housed both via URLs and inside the triple store. This is to facilitate easier sharing and reuse. The submitted prov doc can be shared via the URL for users not familiar with SPARQL or for users who want to consume the full PROV document for other purposes.

**Working with the triple store**

The /exec directory has bash scripts for inserting and deleting files in Virtuoso. These scripts support multiple encodings and are called from within Provisium via os/exec. These bash scripts do need to be updated upon first use with the Virtuoso username and password.

**Sample Command Line Calls Used for Testing**

This section is solely for documenting purposes. To record the syntax of command line calls used when testing Provisium.

(works)

curl -H "Content-Type: text/plain" -d "prov=testValue&key=1234&encoding=turtle" -X POST 'http://localhost:6789/api/v1/prov/submission'

(fails, incorrect number of inputs)

curl -H "Content-Type: text/plain" -d "prov=testValue&key=1234" -X POST 'http://localhost:6789/api/v1/prov/submission'

(works)

curl -H "Content-Type: text/plain" -d "key=1234&password=1234&name=Test Value" -X POST 'http://localhost:6789/api/v1/api/newKey’

(fails, incorrect number of inputs)

curl -H "Content-Type: text/plain" -d "prov=testValue&key=1234" -X POST 'http://localhost:6789/api/v1/prov/update'

**To Do (Not Yet Supported)**

1. There is currently no validation of the PROV documents. I’d like to include a call to Java and Jena for validation. Java is chosen so that we can also include 2.
2. Include this SHACL library to determine dialect of incoming PROV - <https://github.com/TopQuadrant/shacl/tree/master/src/main/java/org/topbraid/shacl/validation>