SPARQL Update API

Purpose

Permits external applications to add or remove specific triples from the VIVO data model. These changes use the standard data channels in VIVO, so the search index will be updated as appropriate, and the reasoner will add or remove inferences as needed.

By default, the SPARQL Update API is disabled in VIVO, for security reasons. See Enabling the API.

Use Cases

Harvester

Previous implementations of the Harvester and similar tools have written directly to the VIVO triple-store, bypassing the usual data channels in VIVO. After ingesting, it was necessary to rebuild the search index, and to run the reasoner to add or remove inferences. Since the search index and the reasoner were not aware of the exact changes, the entire data model was re-indexed and re-inferenced.

When the Harvester and other tools have been modified to use the SPARQL Update API, VIVO will ensure that the search index and inferences are kept in synchronization with the data.

Other ingest tools

This API permits ingest tools such as Karma to programmatically insert data into VIVO without requiring knowledge of VIVOs internal data structures.

VIVO "face" applications

Linked Open Data requests have permitted people to write Drupal applications (for example) that display data from VIVO. This API will permit such applications to accept user edits, and apply them back to VIVO.

Specification

URL

```
[vivo]/api/sparqlUpdate
```

Examples:

```
http://vivo.cornell.edu/api/sparqlUpdate
```

```
http://localhost:8080/vivo/api/sparqlUpdate
```

HTTP Method

The API supports only HTTP POST calls. GET, HEAD, and other methods are not supported, and will return a response code of `405 Method Not Allowed`.

Parameters

name	value	notes
`email`	the email address of a VIVO administrator account	
`password`	the password of the VIVO administrator account	
`update`	A SPARQL Update request	Optional. If used, request content type must be set to `application/x-www-form-urlencoded` and the update data must be URL-encoded.

The syntax for a SPARQL Update request is described on the World Wide Web Consortium site at http://www.w3.org/TR/2013/REC-sparql11-update-20130321/

Limitation

The API requires that you specify a GRAPH in your SPARQL update request. Insertions or deletions to the default graph are not supported.

Sending data via the POST body vs the 'update' parameter

The API supports sending data via an 'update' parameter. Prior to v1.13.0, this was the only method of sending SPARQL update data. Since 1.13.0, VIVO also supports sending data via the POST message body. To use this method, you must set the request content type to `application/sparql-update`. This method parses the data as an input stream and may offer a performance improvement for large amounts of data.

Response Codes

Code	Reason
`200 OK`	SPARQL Update was successful.
`400 Bad Request`	HTTP request content type was set to `application/x-www-form-urlencoded` and did not include an `update` parameter.
	The SPARQL Update request did not specify a GRAPH.
	The SPARQL Update request was syntactically incorrect.
`403 Forbidden`	HTTP request did not include an `email` parameter.
	HTTP request did not include a `password` parameter.
	The combination of `email` and `password` is not valid.
	The selected VIVO account is not authorized to use the SPARQL Update API.
`405 Method Not Allowed`	Incorrect HTTP method; only POST is accepted.
`500 Internal Server Error`	VIVO could not execute the request; internal code threw an exception.

Examples

These examples use the UNIX `curl` command to insert and delete data using the API.

Insert example

This example inserts a single RDF statement into the data model.

Insert via the POST body

```
curl -i --request POST 'http://localhost:8080/vivo/api/sparqlUpdate?email=vivo_root@mydomain.
edu&password=Password' \
--header 'Content-Type: application/sparql-update' \
--data-raw 'INSERT DATA { GRAPH <http://vitro.mannlib.cornell.edu/default/vitro-kb-2> {
 <http://test.domain/ns#book1>
          <http://purl.org/dc/elements/1.1/title>
          "Fundamentals of Compiler Design" .
} }'
```

Insert using the 'update' parameter

```
curl -i -d 'email=testAdmin@mydomain.edu' -d 'password=Password' -d '@insert.sparql' 'http://localhost:8080/vivo
/api/sparqlUpdate'
```

insert.sparql

```
update=INSERT DATA {
   GRAPH <http://vitro.mannlib.cornell.edu/default/vitro-kb-2> {
      <http://test.domain/ns#book1>
          <http://purl.org/dc/elements/1.1/title>
          "Fundamentals of Compiler Design" .
    }
}
```

Modify example

This example removes the previous statement, and inserts a replacement.

Modify via the POST body

```
curl -i --request POST 'http://localhost:8080/vivo/api/sparqlUpdate?email=vivo_root@mydomain.
edu&password=Password' \
--header 'Content-Type: application/sparql-update' \
--data-raw 'DELETE DATA { GRAPH <http://vitro.mannlib.cornell.edu/default/vitro-kb-2> {
 <http://test.domain/ns#book1>
          <http://purl.org/dc/elements/1.1/title>
          "Fundamentals of Compiler Design" .
} }

INSERT DATA {
   GRAPH <http://vitro.mannlib.cornell.edu/default/vitro-kb-2> {
      <http://test.domain/ns#book1>
          <http://purl.org/dc/elements/1.1/title>
          "Design Patterns" .
    }
}'
```

Modify via the update parameter

```
curl -i -d 'email=testAdmin@mydomain.edu' -d 'password=Password' -d '@modify.sparql' 'http://localhost:8080/vivo
/api/sparqlUpdate'
```

modify.sparql

```
update=DELETE DATA {
   GRAPH <http://vitro.mannlib.cornell.edu/default/vitro-kb-2> {
      <http://test.domain/ns#book1>
          <http://purl.org/dc/elements/1.1/title>
          "Fundamentals of Compiler Design" .
    }
}
INSERT DATA {
   GRAPH <http://vitro.mannlib.cornell.edu/default/vitro-kb-2> {
      <http://test.domain/ns#book1>
          <http://purl.org/dc/elements/1.1/title>
          "Design Patterns" .
    }
}
```

Delete example

This example removes the modified statement.

Delete via the POST body

```
curl -i --request POST 'http://localhost:8080/vivo/api/sparqlUpdate?email=vivo_root@mydomain.
edu&password=Password' \
--header 'Content-Type: application/sparql-update' \
--data-raw 'DELETE DATA { GRAPH <http://vitro.mannlib.cornell.edu/default/vitro-kb-2> {
 <http://test.domain/ns#book1>
          <http://purl.org/dc/elements/1.1/title>
          "Design Patterns" .
} }'
```

Delete via the update parameter

```
 curl -i -d 'email=testAdmin@mydomain.edu' -d 'password=Password' -d '@delete.sparql' 'http://localhost:8080
/vivo/api/sparqlUpdate'
```

delete.sparql

```
update=DELETE DATA {
   GRAPH <http://vitro.mannlib.cornell.edu/default/vitro-kb-2> {
      <http://test.domain/ns#book1>
          <http://purl.org/dc/elements/1.1/title>
          "Design Patterns" .
    }
}
```

Clear example

If you want to remove a named graph entirely, you can use the SPARQL CLEAR method.

Clear via the POST body

```
curl -i --request POST 'http://localhost:8080/vivo/api/sparqlUpdate?email=vivo_root@mydomain.
edu&password=Password' \
--header 'Content-Type: application/sparql-update' \
--data-raw 'CLEAR GRAPH IRIRef'
```

Clear via the update parameter

CLEAR example

```
curl -i -d 'email=USER' -d 'password=PASSWORD' -d 'update=CLEAR GRAPH IRIRef' 'http://localhost:8080/vivo/api
/sparqlUpdate'
```

Replace IRIRef with the URI of the named graph you want to delete, e.g. <http://localhost/data/people>

Large Files

For large files one can also use the SPARQL LOAD command.

For this, you have to first create the RDF file with the triples that you want to add, and make the file accessible at a URL. In the example below, the RDF file containing the triples is called `data.rdf`, and is available in the root directory of the web server at `myserver.address.xxx`.

Like the previous commands, this one references a data file, in this case called `import.sparql`. That file contains the `LOAD` command which references the actual data.

```
curl -d 'email=USER' -d 'password=PASSWORD' -d '@import.sparql' 'http://localhost:8080/vivo/api/sparqlUpdate'
```

import.sparql

```
update=LOAD <http://myserver.address.xxx/data.rdf> into graph <http://vitro.mannlib.cornell.edu/default/vitro-
kb-2>
```

Increase the default Tomcat `maxPostSize`

By default, Tomcat sets the default maximum of a POST request to 2 megabytes. If you want to increase this to be able to POST larger sets of triples to VIVO, you can use the `maxPostSize` attribute in `server.xml`. The example below would increase the maximum to 10 MB. See the Tomcat documentation for more details.

server.xml

```
<Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
   URIEncoding="UTF-8"
   redirectPort="8443"
   maxPostSize="10485760"/>
```

Advanced Use

Since v1.13.0, VIVO supports the use of the advanced parameters `using-graph-uri` and `using-named-graph-uri`. You could use these parameters to query a graph, while simultaneously creating new triples based on the query results. For example:

```
curl --location --request POST 'http://localhost:8080/vivo/api/sparqlUpdate?email=vivo_root@mydomain.
edu&password=Password&using-graph-uri=http://vitro.mannlib.cornell.edu/default/vitro-kb-2' \
--header 'Content-Type: application/sparql-update' \
--data-raw 'INSERT {
  GRAPH <http://example.com/new-test-graph> {
    ?s <http://example.com/test> "using-graph-uri worked!" .
  }
}
WHERE
  {  ?s  a <http://purl.org/ontology/bibo/Journal> }'
```

would query the kb-2 graph for subjects of type bibo:Journal, then insert a new triple into the http://example.com/new-test-graph graph based on the resulting subject URIs. Some discussion on using these parameters can be found on this Stack Overflow page.

Enabling the API

Before enabling the SPARQL update handler, you should secure the URL `api/sparqlUpdate` with HTTPS. Otherwise, email/password combinations will be sent across the network without encryption. Methods for securing the URL will depend on your site's configuration.

By default, the SPARQL Update handler is enabled for only the root user in VIVO. To enable it for other user groups, you can either:

- uncomment the line references "UseSparqUpdateAPI" in [vitro]/rdf/auth/everytime/permission_config.n3 or
- create an RDF file in the `[vitro]/rdf/auth/everytime` directory that will authorize your site administrators to use the API. Below is an example of such a file, using N3 syntax.

authorizeSparqlUpdate.n3

```
@prefix auth: <http://vitro.mannlib.cornell.edu/ns/vitro/authorization#> .
@prefix simplePermission: <java:edu.cornell.mannlib.vitro.webapp.auth.permissions.SimplePermission#> .

# Authorize the ADMIN role to use the SPARQL Update API
auth:ADMIN auth:hasPermission simplePermission:UseSparqlUpdateApi .
```