The SPARQL Update API

Remote applications can perform SPARQL Update calls to add RDF to VIVO, or to remove existing RDF. Since VIVO 1.6.

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
`email`	the email address of a VIVO adminstrator account
`password`	the password of the VIVO administrator account
`update`	A SPARQL Update request

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.

Response Codes

Code	Reason
`200 OK`	SPARQL Update was successful.
`400 Bad Request`	HTTP request 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.

```
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.

```
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.

```
 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" .
    }
}
```

A more complex deletion

This update removes an Email address from a Person in VIVO. The Person is related to a VCard:ContactInfo object, which is related to a VCard:EMail object, which has the email address in a data property.

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

delete_email.ru

```
update=
PREFIX obo:      <http://purl.obolibrary.org/obo/>
PREFIX vcard:    <http://www.w3.org/2006/vcard/ns#>

DELETE {
  GRAPH <http://vitro.mannlib.cornell.edu/default/vitro-kb-2> {
    ?contactInfo vcard:hasEmail ?emailObject .
    ?emailObject ?p1 ?o .
  }
} WHERE {
  GRAPH <http://vitro.mannlib.cornell.edu/default/vitro-kb-2> {
    <http://vivo.mydomain.edu/individual/n4295> obo:ARG_2000028 ?contactInfo .
    ?contactInfo vcard:hasEmail ?emailObject .
    ?emailObject vcard:email "my.primary@email.com"^^<http://www.w3.org/2001/XMLSchema#string> .
    ?emailObject ?p1 ?o .
  }
}
```

Big Files

For big 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 make it accessible at a URL that VIVO can access. In the example below, the RDF file containing the triples are the data.rdf (available in the root directory of the web server that respond by the address myserver.address.xxx) and the sparql file below contains the LOAD command to be execute by the SPARQL VIVO API.

import.sparql

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

The command should be the same used in the last examples where USER is a user email authorized to access the SPARQL API (see below) or the email of the root user (created during the install process). The PASSWORD.

command

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

A Python example

Ted Lawless of Brown University has created a Python program to illustrate the SPARQL Update API. You can find it here: https://gist.github.com/lawlesst/6300573#file-vupdate-py

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 disabled in VIVO. To enable it, you must create an RDF file in the `[vivo]/rdf/auth/everytime` directory that will authorize your site administrators to use the API. Here is an example of such a filem 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 .
```

Comment: Actually, this is not 100% true. The root user can always use the SPARQL API and the file `[vivo]/rdf/auth/everytime/permission_config.n3` already contains the line above commented. One only need to uncomment this line to enable the API.