Supporting the Semantic Web and Linked Data

The REST API, Fedora resource identifiers, The Resource index, Named Graphs and the semantic web

Note: The scope of this document goes some way beyond the London Committers' meeting agenda item - Interfaces. Comments welcomed on which are the most relevant issues to cover in the meeting.

This is an attempt to capture this thread

Presentation given at the London 2010 Committer's meeting is here

Fedora in the context of the Semantic web and Linked Data

The basic premise of the proposals below is to support exposing Fedora resources and their relationships in a Semantic Web and Linked Data friendly way. It attempts to "unify" the identifiers and relationships used for Fedora resources with the new REST API and the resource index.

To be Semantic Web and Linked Data friendly involves

- publishing dereferenceable http URIs for resources
- publishing of relationships between resources using these identifiers

The new REST API is a move forward in supporting these requirements as we now have dereferenceable http URIs for Fedora resources.

What this proposal is not about

- Implementation of a semweb publishing mechanism
    - "What" a Fedora object/datastream URI identifies depends on how an individual repository is built
        - Compound vs Complex objects
        - Aggregations, collections
- Not an implementation of OAI-ORE
- But it provides mechanisms to support these

Current situation

- Identifiers
    - Fedora resources have identifiers such as namespace:pid and namespace:pid/datastream, and their info:fedora/ URI forms (and similarly for disseminations)
        - These identifiers are effectively scoped to a repository installation
    - The new REST API provides globally dereferenceable http URIs for resources, but these are not "defined" as (canonical) identifiers for resources.
    - The existing "LITE" APIs also provide resolvable URI resource identifiers
- Relationships
    - The resource index is a single "graph" containing relationships for all objects
    - Relationships must have either a Fedora object or datastream as the subject
        - Limits metadata expression to "flat" schemes such as DC
    - No support for "arbitrary" RDF datastreams in the resource index (eg for implementing additional RDF metadata schemes)
    - Resource identifiers used in relationships are of the info:fedora/ form
        - Difficult to "interpret" relationships outside of the scope of the repository
    - The "specification" of what relationships exist for an object is defined in imperative code

Some Principles

Some basic principles that should be followed by the recommendations below:

- The definitive source of all information should be in Fedora objects
    - No direct manipulation of the triple store - the triple store is a cache/index
- The info:fedora URI scheme should be retained
    - supports re-use of objects across different repositories
- As far as possible Fedora should "work" without using a resource index.

Proposals

1 Deprecate the "LITE" APIs.

- Implement by using HTTP status code 301: moved permanently in the next release
- Remove in subsequent release

2 Define canonical dereferenceable URIs for Fedora resources

- Using the new REST API URIs

3 Restructure the resource index as named graphs

A named graph is a set of triples named by a URI.

For instance, the relationships contained in the object myns:somepid could be identified as the graph <#myns:somepid>. Similarly the relationships expressed by the datastream myns:somepid/RELS-EXT could be identified as <#myns:somepid/RELS-EXT>.

Triple query languages such as SPARQL and iTQL support queries across multiple graphs. Using this to query relationships over the repository as a whole would be complex - it would be painful to have to assemble a list of named graphs to query against.

Mulgara is a quad store, relationships are effectively stored as <graph> <subject> <predicate> <object>. Currently all triples are stored in a single <#ri> graph.

Mulgara supports creating models (graphs) that are views of other models (graphs), eg

- <#myns:somepid/RELS-EXT> containing RELS-EXT relationships
- <#myns:somepid/DC> containing Dublin Core relationships
- <#myns:somepid/properties> containing relationships created from object properties
- <#myns:somepid> as a view defined as the union of the above - this view "contains" all of the relationships for the object
- A view could be created for all relationships in the repository, as a union of all individual object views. This would be equivalent to the current <#ri> graph.

Thus, a hierarchy of named graphs could be created, for example:

- <#ri> - a view containing:
    - <#some:pid> - object graph for some:pid, a view containing:
        - <#some:pid/properties> - graph containing object properties
        - <#some:pid/datastreams> - a view containing:
            - <#some:pid/datastreams/rels-ext> - graph containing triples from rels-ext
            - <#some:pid/datastreams/rels-int> - graph containing triples from rels-int
            - <#some:pid/datastreams/dc> - graph containing triples from DC
            - <#some:pid/datastreams/{rdf datastream}> - graph containing triples from some other rdf datastream
            - <#some:pid/datastreams/{dsid}/properties> - graph containing properties of datastream {dsid} (state, last modified, etc)
    - <#some:otherpid> - object graph for some:otherpid, a view containing:
        - <#some:otherpid/properties> - etc
        - <#some:otherpid/datastreams> - etc

All graphs should be "rooted" in the above structure, there should be no means of creating graphs other than by creating objects and datastreams.

Why do this?

When an object is created (updated, deleted), the object's relationships are propagated to the triple store. If two objects are created expressing an identical relationship, a single triple will be created in the resource index.. If one of those objects is then deleted, the triple will be deleted from the triple store even though it is still being asserted by another object. The resource index will not be an accurate reflection of the triples in the repository. Hence the current restrictions on RELS-EXT and RELS-INT that subjects must be the Fedora object or datastreams from the containing object, to prevent two objects asserting the same relationship.

With named graphs, relationships created by different objects would be in different graphs. Deleting one object would remove the graph for that object - but the graph for a different object asserting the same relationship would remain - the resource index would be an accurate reflection of the triples in the repository.

Therefore this would support indexing of arbitrary RDF metadata datastreams in the resource index - for instance supporting metadata schemes that are not "flat".

There are some Dublin Core examples [DEV:1] where Fedora would currently be unable to index the RDF in the resource index, including

- using foaf to describe a person who is a dc:creator of a resource
- identifying the taxonomy used to populate dc:subject

[DEV:1] http://dublincore.org/documents/dc-rdf/#app-a

Questions and issues

- The graph hierarchy to use - how granular? Start with something simple?
- Mapping between resource identifiers and graph names
- Separation of "core" relationships from "user-defined" relationships into different overall views? If the intention of the resource index is to store relationships between objects, we may not want to pollute that with other relationships, eg from arbitrary RDF datastreams
    - Relationships about the object and its datastreams - in <#ri>
    - Relationships from RELS-EXT, RELS-INT, DC - in <#ri>
    - Relationships from arbitrary RDF datastreams/disseminations - in <#riUser>
    - <#riFull> as a union of <#ri> and <#riUser>
- Performance. Need to evaluate query performance over a network of named graphs vs storing all relationships in one single graph

- Triple store support: Mulgara supports named graphs and views, what about other triple stores? MPTStore?
- Impact on Mulgara's free-text index, do we create a parallel structure of free text graphs? Does Mulgara even support this?

4 Declarative specification of triples to create in the resource index

Triples are currently created for

- object properties
- datastream properties
- reserved datastreams that contain RDF (RELS-*)
- reserved datastreams are translated to RDF (DC)
- relationships between objects and their datastreams and disseminations
- relationships between objects and their content models

The "specification" of what triples get generated is largely in imperative Java code, both in terms of the individual triples and which datastreams generate triples.

In the future we may wish to allow creation of triples from

- arbitrary RDF datastreams
- arbitrary XML datastreams to be "lifted" to triples
- disseminations serving RDF

To support a flexible and extensible approach, we could define the generation of triples using content models (system and user) and a declarative approach for specifying triples (XSLT, GRDDL[DEV:1]).

- System content model disseminators for generating RDF for
    - Object and datastream properties triples (from the object's serialisation/FOXML)
    - Relationships between objects, datastreams and disseminations (from the object's serialisation/FOXML)
    - XML datastreams (DC)
- User content models specifying
    - Additional arbitrary RDF datastreams to index
    - RDF disseminations to index
    - Conversion patterns for other XML datastreams and disseminations

Updating of the resource index could then take place by querying the disseminations and datastreams specified by the system and user content models when an object is created, updated or deleted.

The resource index is currently updated by code in DOManager. An alternative to this could be to reimplement the update mechanism using a management decorator pattern (declared in fedora.fcfg).

[DEV:1] GRDDL is a mechanism for Gleaning Resource Descriptions from Dialects of Languages. It is a technique for obtaining RDF data from XML documents and in particular XHTML pages: GRDDL Primer http://www.w3.org/TR/grddl-primer/

Questions and Issues

- How to define the above using system and user content models
- How to specify the mapping between XML and RDF

5 Extend the REST API to incorporate relationships

The REST API does not currently implement methods for disseminating and managing relationships.

API methods should be implemented for querying and managing relationships.

For example

- GET /objects/{pid}/relationships - return RDF for all relationships for the object
- GET /objects/{pid}/datastreams/DC/relationships - return RDF for the DC datastream

Alternatives to explicit "relationships" URIs could be

- Use content negotiation, eg Accept application/rdf+xml and use the existing REST URIs
- Use a "format" URL query string, eg format=rdf
- Or both...

Modifications could be specified by

- POST a set of triples to create new ones
- DELETE a set of triples to be deleted
- PUT a set of modifications to perform, eg using (a subset of) SPARQL Update [DEV:1]

Additionally, or alternatively, "writeable methods" could be provided as a generic mechanism to implement this, eg PUT a SPARQL Update to /objects/{pid}
/methods/{sDefPid}/relationships?datastream=RELS-EXT

All of the relationship API methods should operate directly on Fedora objects to remove dependency on the resource index - relationship GET methods should query the object directly rather than issuing RI queries.

[DEV:1] SPARQL Update - A language for updating RDF graphs: http://www.w3.org/Submission/SPARQL-Update/

Questions and issues

- REST endpoints to use - explicit relationships URIs vs content negotiation vs URL query string
- Relationships update specification (SPARQL Update, or ...)
- Supporting "generic" updates, eg repository-wide relationships methods and methods operating on an object as a whole
  - Subject and predicate can be used to determine what to update for object properties, datastream properties, Dublin Core
  - RELS-EXT, RELS-INT and arbitrary datastreams present a challenge. A triple with a Fedora object as a subject could be stored in RELS-EXT or in an arbitrary RDF datastream. Do we restrict fedora-model and fedora-system predicates to RELS-EXT and RELS-INT?
- Supporting updates to XML datastreams that get converted to RDF
  - eg updating DC through relationship API methods

6 Support for dereferenceable http URI resource identifiers in relationships

Fedora resources are currently identified using the info:fedora namespace. If resource identifiers are exposed as dereferenceable http URIs using the REST API URIs, it would be useful to support these identifiers in relationships. Ie the ability to query and manipulate relationships using both the info:fedora namespace for Fedora resources and the http REST URIs.

REST API

- Provide the ability to query and manipulate relationships using the REST API http URIs
  - Maybe a URL query string parameter? scope=local for info:fedora, scope=global for http URIs?

RISearch

- Query using either info:fedora URIs or the REST API http URIs
- Return results using either info:fedora URIs or the REST API http URIs
- Some form of query re-writing and result set rewriting?
- RISearch query string parameter to determine the form of identifiers to use?

A Spanner (Wrench) in the works...

Fedora repositories generally sit behind some form of user interface application.
These applications will (in some cases) expose their own URLs for accessing Fedora resources
Should we instead be providing mechanisms to support exposing these URLs as the canonical http URIs for Fedora resources?