Managing Semantic Relationships in RUcore Release R5.2

Introduction

We have concluded in earlier architecture discussions that several upcoming RUcore applications will use Fedora's relationship services. At this point, there are four possible applications: annotations, ETD compound objects (i.e. with supplementary files), data compound objects and a service for administrators/collection managers. This document proposes that we proceed in R5.2 to support relationships for annotations and a related management function that would allow an administrator to create, edit, or delete relationships. This approach will give us experience with Fedora relationships and also allow us to experimentally create and evaluate compound objects for the data portal and ETDs. Both the annotation application and the management function should utilize a generic API service layer for managing Fedora RELS-EXT relationships. A proposal for this service layer is detailed in the last section of this document.

Annotations

From the review of the annotation specification, the following assumptions are made:

- The annotation application creates the relationships (hasAnalytic/isAnalytic) when the annotation object is created.
- When an annotation is deleted, the annotation application will delete the relationship in the resource object.
- A single object can possibly have many annotations.
- The resource object will show all the annotations in the full record display.
- Search results will only show the resource object although the metadata for the annotation objects will be indexed as part of the resource object.
- Annotations are included in the collection to which the resource belongs.

With the above assumptions and the more simplified approach in R5.2, it will not be necessary for WMS to make any changes in R5.2 to support relationships, i.e. the annotation application will manage the relationships and any special purpose editing can be accomplished the management function in dlr/EDIT. It is assumed that an annotation object will be created according to WMS conventions and therefore could be edited by WMS.

Management Utility (dlr/EDIT)

It is expected that there will be occasions to create, edit, and delete relationships between objects. It is proposed that an administrative tool be created that would provide the following capabilities:

- Browse functions. Show all the pairs of objects in a collection that have the XXX/YYY relationship. From this list, the user can select a pair for editing of relationships.
- The user selects two objects that have relationships. This might be done by searching and selecting two objects from search results or perhaps the user has determined the Fedora IDs in some other process.
- Given that two objects have been identified, the user is presented with a display that has a picklist of all the currently available relationships in RUcore. Upon selecting a relationship, this feature would create both forward and backward pointing relationships.
- The user can also:
    - Delete relationships
    - Add new (possibly multiple) relationships
    - Verify that the relationships have been added properly
- Although we are planning to use only the Fedora-supplied ontologies in R5.2, we will likely need a capability to create new ontologies in the future.
- Support of the ontology registry (see Architecture below).

Relationship Service API Architecture

The concept is to add a RESTful service layer on top of built in Fedora API functionality for managing RELS-EXT relationships. This would require the creation of four new methods under a new service called 'relationship.'

Methods

relationship/add – This method would be used to create a REL-EXT statement between two objects.

Required parameters

Host Fedora Object ID

Fedora Object ID of related object

Relationship ontology to use

Reference: http://www.fedora-commons.org/documentation/3.0b1/userdocs/server/webservices/apim/index.html#methods.addRelationship

relationship/delete – This method would be used to delete a REL-EXT statement that exists between two objects.

Required parameters

Fedora Object ID

Datastream ID

Relationship ontology to delete

Reference: http://www.fedora-commons.org/documentation/3.0b1/userdocs/server/webservices/apim/index.html#methods.purgeRelationship

relationship/get – This method would return the relationships used present in a given fedora object

Required parameters

Fedora Object ID

Reference: http://www.fedora-commons.org/documentation/3.0b1/userdocs/server/webservices/apim/index.html#methods.getRelationships

relationship/replace – This method would replace an established relationship with a new ontology by using a combination of Delete and Add methods

Required parameters

Host Fedora Object ID

Datastream ID

New Relationship ontology to use

Relationship ontology to replace

All methods would only be accessible to by applications that have a valid access key which is also passed at time of interaction with any of the methods. This will provide an access control layer between this service and the repository.

Application Access Control

Access will be provided by using a database in the backend to control application access to the methods. Below is a preliminary table structure that could be used for this purpose.

Key	**Add**	**Delete**	**Get**	**Replace**

An example of its usage might be

Key	**Add**	**Delete**	**Get**	**Replace**
WMS	TRUE	TRUE	TRUE	TRUE
Search	FALSE	FALSE	TRUE	FALSE

Analytic	TRUE	TRUE	TRUE	TRUE

In this example the applications WMS and Annotation have full access to all functionality, whereas Search can only query for relationships that are present within a single object.

Ontology Registry

Ontologies are provided with Fedora, http://www.fedora-commons.org/definitions/1/0/fedora-relsext-ontology.rdfs, however we will find the need to create ontologies of our own at some point as well. Also certain actions will need to be taken depending on ontologies used in a RELS-EXT statement. In order to mange this all methods outlined above will need to refer to a database table for guidance. A preliminary structure might be the following.

ID	Prefix	Term	Definition	FollowGet	FollowDelete

An example of its usage might be

ID	Prefix	Term	Definition	FollowGet	FollowDelete	IndexIndependent
1	Has	Analytic	Describes the object has an object that is annotating it	TRUE	TRUE	TRUE
2	Is	Analytic	Describes the object is annotating another	TRUE	FALSE	FALSE

In this example when an object is using the "hasAnalytic" ontology the table, FollowGet, says when using the GET method, follow the RELS-EXT statement to obtain what it has annotated. In the event this object is purged the FollowDelete says to follow the REL-EXT statement and purge the related object. Finally, IndexIndependent states that the object using this ontology can be indexed independently.
For the "isAnalytic" example when using the GET method on an object with this ontology it will follow that relationship. When purging this object however the related object setting FollowDelete to FALSE will say to not delete the related object. Finally IndexIndependent set to FALSE says to not index the object on its own.

rcj/cmm/jat – 05/12/2010