Sketching out a read+write REST-friendly CMA

Introduction

The current Fedora Content Model Architecture (CMA) achieves at least two great feats:

1. Describes the physical characteristics of objects in a given model. Currently, this comprises an enumeration of required Datastreams and their properties (e.g. datastream ID, mime type, format URI). This is entirely new in Fedora 3.0.
2. Enumerates a set of fedora-mediated services available for all objects in given mode (e.g. entire classes at a time), and binds these services to their implementation. Prior to Fedora 3.0, these services (and their bindings) were explicitly enumerated in each and every every object instance.

Current state & mental model

Here, we are concerned with the services aspect of the CMA. As they exist now, services in Fedora have the following basic mental model:

```
result = f(object, sDef, method, parameters)
```

where `result` is a stream of bytes, `object` is the identity (pid) of an object, `sDef` is the identiy (pid) of a Service Definition object that is related to one of the object's content models via the `fedora-model:hasService` relationship, `method` is the name of an operation enumerated in `sDef`, and `parameters` are a series of constrained key/value pairs.

Interfaces: SDefs and fedora APIs

Service Definitions (SDefs) enumerate methods, as well as parameters for each method. In an SDef, parameters may be given a default value, or have the set of the set of possible values enumerated. At present, the methods and parameters used when invoking a service MUST be defined in the SDef - any parameters or methods not explicitly declared in the SDef are always invalid.

In reality, the two means by which services may be invoked are via API-A and API-A-Lite.

In API-A, using SOAP:
`getDissemination(object, sDef, method, parameters)`

and in API-A-Lite, using HTTP GET:
`http://fedora.base/get/\{object}/{sdef}/{method}?{parameters}`

There is no way to interact with services in the current experimental REST/HTTP API - that part has not been implemented yet.

Implementation: SDeps and WSDL

Service Deployment (SDep) objects are the non-public component of services in the CMA. They implement the methods described in a given SDef, for a specific set of models. As such, SDep objects themselves declare (a) which SDef they implement and (b) for which content models they implment that SDef. The point here is that one common interface (SDef) may have many different implementations, each one specific to objects of a particular model. When a service is invoked for a given object, Fedora picks the correct SDep based upon the above criteria.

Current SDeps in Fedora implement an SDef by invoking some other web service, passing it parameters, and returing the results. This binding is defined in the WSDL datastream. Therefore, when Fedora is executing an SDef method, it chooses an SDep (as described earlier), and reads its WSDL datastream. Fedora then invokes an external web service based on the specifications set forth in the WSDL datastream. In theory, this binding can be arbitrarily complex and flexible, but in reality Fedora only implements an HTTP GET binding to external services right now. Other HTTP methods, or even soap bindings to external services, for example, are not implemented. Parameters and datastreams (datastream IDs, rather) are therefore passed to the external service by encoding these values in the URL. Once Fedora invokes an external service, its http response body and headers are returned the caller.

Read-only nature

Currently, Fedora services and disseminations have been assumed to be read-only. In theory, an external web service called from an SDep could itself make API-M request to Fedora and modify the repository state, effectively making a particular service read/write. However, in the current state of the dissemination architecture, this would cause some architecturally unpleasant consequences:

1. A user/client invokes services through API-Lite using the http GET method - which according to RFC 2616 (HTTP/1.1) should be assumed safe and idempotent. If a service then alters repository state, this assumption may be violated.
2. Fedora uses http GET to invoke the backend services from SDefs. Again, if the backend service alters repository state, then this service must operate counter to the expectations of http.

The AtomPub thought experiment provides one compellng example for a fully read-write CMA-based service framework, and specifically focuses an HTTP+REST mode of interaction - something which is currently not supported in Fedora. From the AtomPub example, we see a few obvious features that are missing:

1. The ability to use HTTP methods beyond 'GET'
2. The ability to process HTTP headers as input (eg. the `Slug:` header in AtomPub)

Can these features be added to Fedora's APIs without violating the current service mental model? Do these features have to exist within the current mental model at all?

1. There is no way to distinguish idempotent from safe from unsafe operations in the current model
    a. Up until now, service operations have been assumed to be safe and idempotent. A read+write API will invalidate that assumption
2. HTTP methods are different from `methods` in the service model
    a. Mapping HTTP methods to method parameters is technically possible, but seems less than ideal. For example, POST map to parameter `http.method=post`
3. HTTP headers, as inputs, are not directly supported
    a. Mapping HTTP headers to input parameters is technically possible. For example, `Slug: XYZ` may map to `Slug=XYZ`

(to be continued)