Notes on SDef and SDep Improvements

Proposal notes for London Committers' Meeting, Feb. 23-24, 2010

The four broad areas of discussion under this proposal are:

1. Generally localized changes in service resolution and improved documentation, with minimal disruption to the existing implementation
2. More significant changes to the use of WSDL in service resolution, including additional http binding styles and verbs
3. SOAP support, either as a back-end binding to a service deployment or a front-end interface to POST-supporting service definitions
4. An overview of ways to model writeable endpoints, REST, and more flexible integration with various APIs through the SDef/SDep architecture

1. Refinements in the scope of the Fedora 3.3 Service Resolution

Changes to ServiceMapper

More Robust Port-to-Method Bindings

Allow multiple bindings to a service. This involves simply changing the search logic for constructing the MethodDefOperationBind array to search available HTTP bindings for an operation name (defaulting to the first), rather than defaulting to the first HTTP binding for all operations. This aligns with canonical documentation for WSDL services, creates an opening to distinguish bindings by verb, and allows for less cumbersome markup of services, e.g.:

```
          <wsdl:binding name="risearch_http" type="this:examplePortType">
           <http:binding verb="GET"></http:binding>
           <wsdl:operation name="getSize">
             <http:operation location="/risearch?type=tuples&amp;lang=itql&amp;format=count&amp;query=select%
20%24member%20from%20%3C%23ri%3E%20where%20%24member%20%3Chttp%3A//purl.oclc.org/NET/CUL/memberOf%3E%20%3C
(objuri)%3E"></http:operation>
             <wsdl:input>
               <http:urlReplacement></http:urlReplacement>
             </wsdl:input>
             <wsdl:output>
               <mime:content type="text/plain"></mime:content>
             </wsdl:output>
           </wsdl:operation>
         </wsdl:binding>
         <wsdl:binding name="lamp_http" type="this:examplePortType">
           <http:binding verb="GET"></http:binding>
           <wsdl:operation name="listMembers">
             <http:operation location="/fedora-svc/aggregator/listMembers/bag-aggregator.php?nullbind=
(NULLBIND)&amp;objuri=(objuri)&amp;callback=(callback)"></http:operation>
             <wsdl:input>
               <http:urlReplacement></http:urlReplacement>
             </wsdl:input>
             <wsdl:output>
               <mime:content type="text/xml"></mime:content>
             </wsdl:output>
           </wsdl:operation>
         </wsdl:binding>
         <wsdl:service name="aggregator">
           <wsdl:port binding="this:risearch_http" name="risearch_port">
             <http:address location="http://local.fedora.server/fedora"></http:address>
           </wsdl:port>
           <wsdl:port binding="this:lamp_http" name="lamp_port">
             <http:address location="http://php.example.edu"></http:address>
           </wsdl:port>
         </wsdl:service>
```

ServiceDeployment WSDL cannot support relative URIs in http:address

FCREPO-619 is a reasonable expectation given the W3C docs for wsdl . The principal question is what the context for resolution should be.

Define and Implement a Pluggable Interface for the Creation of ServiceMapper Instances

One way to cordon off most any proposed change to service deployment bindings to minimize disruption would be to extract an interface from the few public methods of the ServiceMapper class, re-implement the existing class as an implementation of the interface, and define a factory interface as an extension of Pluggable. This is extending the use of the plugin architecture to mimic SPI:

```
 interface ServiceMapperFactory extends Pluggable {
    public ServiceMapper getServiceMapper(String parentPid);
}
```

```
 interface ServiceMapper  {
    public MethodDef[] getMethodDefs(InputSource methodMapSource)
            throws ObjectIntegrityException, RepositoryConfigurationException, GeneralException;
    public MethodDefOperationBind[] getMethodDefBindings(InputSource wsdlSource, InputSource methodMapSource)
            throws ObjectIntegrityException, RepositoryConfigurationException, GeneralException;
    public DeploymentDSBindSpec getDSInputSpec(InputSource dsInputSpecSource)
            throws ObjectIntegrityException, RepositoryConfigurationException, GeneralException;
}
```

... with the existing ServiceMapper becoming an AbstractServiceMapper (initializers) & ServiceMapperImpl

Implement More InputParm Binding Options

Service definitions still focus on a minimal set of UserInputs, but service deployments might have a richer array of options made available to it. This would require a change to the Service Deployment Method Map document type definitions, and code changes in DefaultAccess to accommodate what are functionally akin to DefaultInputParms.

1. bind to DC element
    - DCInputParm could be marked up similarly to existing parms, with the @parmName value indicating the dc predicate to search for the value on
    - Alternately, an extension to the existing InputParm types that added a @predicate value, whose object would be bound to @parmName
    - Could cause errors if predicates with multiple statements are used in urlReplacement context
2. bind to object of RDF triple from RELS-EXT or RELS-INT
    - As above, requires both @predicate and (if RELS-INT supported) @dsSubject
    - As above for multiple value concerns
3. bind to value of a system or environment property
    - @parmName could map to system property relatively easily
    - Correspondence to String-String hash should eliminate mutiple value concerns

Improving Documentation and Validation Tools

Existing documentation focuses on SDefs and SDeps as Fedora objects, explaining the required datastreams and their expected content. It is less concerned with what SDefs and SDeps are, how they are related. Similarly, runtime errors in service resolution often have opaque, wrapped Null Pointer exceptions, or throw 500's. Finding ways to improve error messages and documentation would make the service resolution architecture much more approachable. It would also be helpful to implement a debugging tool to identify common error patterns in SDef/SDep pairings (maybe even a service definition for SDep objects!)

2. Changes to Parsing and Use of WSDL, Front and Back end

Support for WSDL Faults (API WSDL & SDef/p)

FCREPO-52 is really a documentation issue. What fault types might there be? org.fcrepo.server.utilities.AxisUtility relies on org.apache.axis.AxisFault for construction; for all subtypes of ServerException, detail is set as a buffer of <detail> elements; for AuthzExceptions, set by qname="Authz" with standard message.
We should be able to define two message formats to accommodate all the API messages.

FCREPO-52: WSDL Should Declare Faults

Support http:urlEncoded

Supporting changes would need to be made to HttpOperationInOut and WSDLParser (to support the value) and DisseminationService. assembleDissemination (to build the URL). This appears to be a less dramatic change than some others, and would offer a way to support multiple-value input parms without WSDL2 support. Along with some of the other proposed features, a key component is probably refactoring DisseminationService into an interface, with supporting implementations for different operation types.

Note: In WSDL-1.1, these http binding options are all-or-nothing. WSDL-2 has a more sophisticated reckoning of serialization style application/x-www-form-urlencoded .

Rudimentary support for HTTP POST

If the ServiceMapper class is refactored to allow multiple bindings, it opens the door to supporting additional http verbs. This would require some additional refactoring of the binding classes to key on verb, and elaborating org.fcrepo.common.http.WebClient to support POST. This is only considering support for the verb (for example, to skirt URL length restrictions on bound services), not its role in REST or the requirement of idempotency.

Deliver Composed WSDL for Object Disseminations

Variant on the ListMethodsServlet calls with xml=true. Ideally, this would parse deployments rather than definitions.

3. SOAP Support

There are two broad areas of SOAP suport that may be pursued: SDep binding to SOAP services, and SDef specification of SOAP services.

Allow SDeps to Bind to SOAP Services

Allowing SDeps to bind to SOAP services allows a Fedora object to mask complex behaviors relying on datastream content and metadata with a relatively simple interface of UserInputParms. It requires support for HTTP POST, and indicates a different context for the use of existing input parms. Much of the scaffolding for SOAP bindings already exist in the codebase, with the significant exceptions of ServiceMapper (which takes no binding actions for SOAP services), DisseminationService (which always assumes http:urlReplacement), and DefaultExternalContentManager (which neglects SOAP, and only support http GET).

Message part definitions include types; would have to assume corresponding order

Example: Document sdef with a getCalais service defined - sdep might have user parms for API key; nullbind-style bind to paramsXML, bind to content datastream

Potential problems: Support for xsd types is easy; support for complex or user-defined types is more difficult. Need to specify some rules for attempted type conversion. Somewhat awkward reliance on Fedora admin to keep SDep wsdl and actual service wsd in synch. Perhaps allow WSDL by reference?

If SOAP binding for SDeps is supported, how will faults be handled? Expected/unexpected?
Does MIMETypedDatastream need to be given a way to communicate response codes? How should ExternalContentHandler communicate fault information?

Support mime:multipartRelated

Necessary to support SOAP with attachments or MTOM, allows for some transport efficiency of b64 encoding.

Allow SDefs to Specify SOAP Services

The principal advantage is probably a follow-on to SDep SOAP binding: It would prevent the context object from being responsible for all data for a bound service. It introduces an additional complication in having to translate the client input document into an appropriately formatted input for the bound service. There is also the question of how to constrain message and fault types practically.

4. Strategies for services as different types of endpoints

WSDL-2 Support in Service Definitions and/or Deployments

Pros: Better language for multiple verbs; better support for serialization of multiply-valued parts into URLs for GET requests or XML for POST; request header support; multipart/form-data support; supported by Axis2

Cons: People kind of hate it; changed characters for value substitution mean patterns aren't backward compatible; questionable whether it's necessary for desired REST-like support

It woud be an open question whether the relatively rigid delineation of paths suits the needs of some previously indicated support for better mapping of external APIs to service definitions (eg FCREPO-405).

It might very well be flexible enough to model a known external API, so there's a possible path of defining WSDL-2 SDeps, and having them proxied by a simpler WSDL-1.1 SDef

See also:

- IBM developerWorks article on REST and WSDL-2.
- XML.com on WSDL-2

FCREPO-500: Writeable disseminators; POST support

Writeable disseminators are more than just post support- they suggest a view of the endpoint as a resource rather than a messaging address.

This is a problem considered in Asger's REST API proposal and in Aaron's sketch of a Read Write CMA .

FCREPO-500

Other Things to Chew On:

A thread in which a savvy developer doesn't want to deal with SDef/p

---

Proposal Outline

1. Refinements in the scope of the Fedora 3.3 Service Mapping
    a. Tweaking the ServiceMapper class

        i. More robust port-to-method binding
        ii. Define and Implement a pluggable interface
        iii. FCREPO 619: ServiceDeployment WSDL cannot specify relative URIs in http:address
    b. More input parm binding options
        i. bind to DC element
        ii. bind to object of RDF triple
        iii. bind to a system or environment property
    c. Improving documentation and validation
        i. Documenting the role and relationship of similar structures in SDef/SDep
        ii. A trouble-shooting/analysis tool to identify problems (perhaps part of an EZService bundle)
2. WSDL Descriptions: Front-end and back-end
    a. WSDL 2 support
    b. Delivery of composed WSDL for clients at service endpoints
    c. FCREPO-52: WSDL Should Declare Faults
    d. FCREPO-500 ; writeable disseminators, REST-as-service, POST support for (hopefully) idempotent services?
3. FCREPO-16: SDeps with SOAP Bindings
    a. Back-end with Datastreams (SOAPInputParm? DSInputParm@passBy='VALUE'?)
    b. Front-end as service (follow-on to FCREPO-500?)

3.2 Notes

1. Only supports one wsdl:service definition (contra WSDL 1.1), multiple wsdl:service elements will override previous
    - This is dependent on the WSDLParser, which only allows a single service; as well as the ServiceMapper, which only expects one
2. FCREPO-16: Does not support SOAP message bindings
3. Effectively limited to one port (operations are all mapped to the first port with an HTTP binding)
    - see ServiceMapper.merge();
    - Consequence: Operations can't be grouped according to http:address, so different service hosts mean only LOCAL addressing can work
4. FCREPO 619: ServiceDeployment WSDL cannot specify relative URIs in http:address
    - Does allow local.fedora.server variable as location
5. Effectively limited to a single portType (WSDLParser has one Service, Service class has one PortType)
6. Appears to require all portTypes to include all abstract message types? (still looking into this)

Wishlist

1. Allow multiple ports and port types
2. Clarify documentation on meaning of required parms and default values in SDef and SDep