# Re-Implementing Service Deployment

This page is to follow up on a number of conversations at the 2010 London Committers' Meeting relating to a re-working of the service deployment implementation under the Fedora 3.x CMA.

The consensus at the meeting was that the existing implementation of service deployments left much to be desired: It's implementation of WSDL-defined services is partial, and WSDL itself is a strained choice for describing many of the actual service deployments, which are more likely to be simple HTTP services or REST endpoints.

Desiderata included:

- Support for a much simpler description language for http services, cf. Asger's Datastream Methods
- Support for SOAP service bindings
- Support for additional HTTP verbs
- REST-style endpoint support
- Tooling and Documentation improvements to support authoring and debugging of service deployments
- More complete WSDL support, for deployments using WSDL

Broadly, we want to sketch out two categories of proposals: One in that works in the 3.x CMA and doesn't invalidate existing SDep objects, but allows for entirely different types to be deployed, and (if necessary) one that is targetted at a 4.x CMA with only a consideration for translation of 3.x objects.

## 3.x CMA Sketch

### SDep Object Definition and Markup Considerations

For a hypothetical service deployment <hypothetical:SDep>, there are a number of triples encoded in the object that make it actionable:

```
<hypothetical:SDep-3.0> <fedora-model:hasModel> <info:fedora/fedora-system:ServiceDeployment-3.0>
<hypothetical:SDep-3.0> <fedora-model:isDeploymentOf> <hypothetical:ServiceDefinition>
<hypothetical:SDep-3.0> <fedora-model:isContractorOf> <hypothetical:ContentModel>
```

... with no defined or effective restrictions on the cardinality of the latter two.

To allow for new service deployment mechanisms without invalidatng existing service deployments we might introduce a new content model:

```
<info:fedora/fedora-system:ServiceDeployment-3.0> <fedora-model:hasModel> <info:fedora/fedora-system:
ServiceDeploymentType>
<hypothetical:SDep-3.0> <fedora-model:hasModel> <info:fedora/fedora-system:ServiceDeployment-3.0>
<hypothetical:SDep-3.0> <fedora-model:isDeploymentOf> <hypothetical:ServiceDefinition>
<hypothetical:SDep-3.0> <fedora-model:isContractorOf> <hypothetical:ContentModel>
```

... such that we infer that <hypothetical:SDep-3.0> is a service deployment from its deployment and contractor statements, and determine the type of service deplyment from its content model when said model has itself the <info:fedora/fedora-system:ServiceDeploymentType> model. This leaves all existing service deployment objects unchanged, but alows for the introduction of new types.

( type issue / inheritance of cmodel; whether inference is appropriate )

### SDep Interface and Code Considerations

#### Relevant Pluggable Interfaces

##### org.fcrepo.server.access

- Access
- DynamicAccessModule: Not directly involved in service resolution, but may offer a model for impl loading

##### org.fcrepo.server.storage

- DOManager

#### Relevant "Private" Interfaces

##### org.fcrepo.server.access.dissemination

- DisseminationService
    - Instantiated by DefaultAccess, with a parameter ServiceDeploymentReader, and DatastreamResolverServlet for (basically) access to static members/methods

### org.fcrepo.server.storage

- DefaultDOManager
  - Redundant implementation of methods for getServiceDefinitionReader; getServiceDeploymentReader
- DirectoryBasedRepositoryReader
  - Redundant implementation of methods for getServiceDefinitionReader; getServiceDeploymentReader
  - Is this in use outside of its test classes?
- SimpleServiceAwareReader
  - Does this need to extend DOReader?  It could easily have a DOReader member.
- SimpleServiceDefinitionReader
- SimpleServiceDeploymentReader

### org.fcrepo.server.storage.service

- ServiceMapper

### org.fcrepo.server.resourceIndex

- ServiceDefinitionTripleReader_3_0
  - Instantiates ServiceMapper to get abstract method defs.  This method is available through SimpleServiceDefinitionReader, and the instantiation is in a private method, so seems refactorable.  Could follow DefaultAccess model and request reader from DOManager.
- ServiceDeploymentTripleGenerator
  - Doesn't do anything!
- ModelBasedTripleGenerator
  - Maps the above to generator classes by RDFName, the map generation is hard-coded

### org.fcrepo.common.http

- WebClient : Only supports HTTP GET

## Cached Lookups and the Internal Db

( Indexing into internal Db; incomplete definitions )

### Proposing New and Refactored Interfaces

( Defining interfaces to allow new implementation / pluggability; Spring/OSGi considerations )

### org.fcrepo.server.service.storage

- ServiceDefinitionReader
- ServiceDeploymentReader
  - ServiceMapper becomes a package-internal mechanism

### org.fcrepo.server.service.storage.impl.default_3_0

- SimpleServiceDefinitionReader (moved from org.fcrepo.server.storage.SimpleServiceDefinitionReader)
- SimpleServiceDeploymentReader (moved from org.fcrepo.server.storage.SimpleServiceDefinitionReader)
- All classes from org.fcrepo.storage.service?

### org.fcrepo.server.service.dissemination

- DisseminationService ( interface derived from existing org.fcrepo.server.dissemination.DisseminationService )
  - Implementation to be provided by a generating method in ServiceDeploymentReader

### org.fcrepo.server.service.dissemination.impl.default_3_0

- DisseminationServiceImpl (moved partially from org.fcrepo.server.dissemination.DisseminationService )

### org.fcrepo.server.service.resourceIndex

- ServiceDeploymentTripleGenerator
  - must provide an indicating cmodel uri
- ServiceDefinitionTripleGenerator
  - may not be necessary yet?
  - must provide an indicating cmodel uri

( Moving all existing resolution / binding logic into an implementation indicated by fedora-system:ServiceDeployment-3.0, defining interfaces and loading faculty for completely different implementations )

## Module Architecture

### DOManager

Assuming the persistence of the Fedora 3.0 plugin architecture for 3.4+: If the existing service resolution functions are better encapsulated behind an implementation of ServiceDeploymentReader provided by DOManager, DOManager becomes the logical place for the steering decision (based on the SDep's content model). One hitch is the resource index rebuild: A DOManager isn't necessarily available to the triple generators. Perhaps DOManager could share a delegate for driving this behavior with DirectoryRepositoryReader?

### Spring Modules

Even in a scenario that preserves existing CMA 3.0 object markup, the proposed change to a model-dependent loading of deployment readers and triple generators depends on the module architecture. If Spring is included in a 3.4 release, then the DOManager is not necessarily the arbiter of cmodel-indicated implementations. Is it reasonable to propose that areas of code that provide one or more implementations based on cmodel are application contexts?

If so, then the separate interface definitions for the SDef and SDep triple generators are probably unnecessary. Instead, we may have an application context for triple generation, one for service definition, and another for service resolution?

Returning to ModelBasedTripleGenerator as an example:

```
<beans>
 <bean id="ModelBasedTripleGenerator"
  class="org.fcrepo.server.resourceIndex.ModelBasedTripleGenerator">
  <constructor-arg>
    <map>
      <entry key="info:fedora/fedora-system:ServiceDefinition-3.0">
        <bean class="org.fcrepo.server.resourceIndex.ServiceDefinitionTripleGenerator_3_0"></bean>
      </entry>
      <entry key="info:fedora/fedora-system:ServiceDeployment-3.0">
        <bean class="org.fcrepo.server.resourceIndex.ServiceDeploymentTripleGenerator"></bean>
      </entry>
      <entry key="info:fedora/fedora-system:ContentModel-3.0">
        <bean class="org.fcrepo.server.resourceIndex.ContentModelTripleGenerator_3_0"></bean>
      </entry>
      <entry key="info:fedora/fedora-system:FedoraObject-3.0">
        <bean class="org.fcrepo.server.resourceIndex.FedoraObjectTripleGenerator_3_0"></bean>
      </entry>
    </map>
  </constructor-arg>
 </bean>
</beans>
```

This works because the triple generators are re-usable. To implement service-aware readers, the model driven behaviors need to provide a factory method of some kind.

(service-aware reader example)

(model-driven behavior example)

## 4.x CMA Sketch

More to come, but presumably moves away from any WSDL specification as the default type.