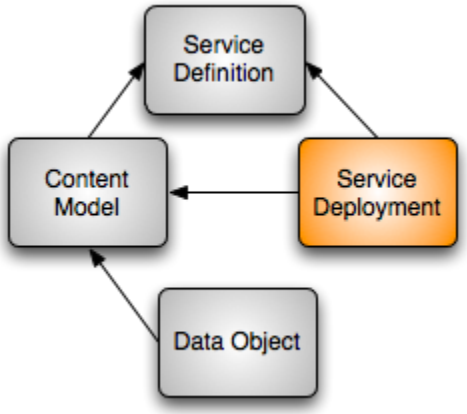
Creating a Service Deployment



This guide describes how *Service Deployment Objects* are constructed in detail. In Fedora, a Service Deployment ("SDep") is a special kind of object that describes how the methods of a particular Service Definition are to be deployed for a particular Content Model.

Although you may author Fedora objects in other ways (e.g. by making a series of API calls to your repository), we will assume for this guide that the object is being constructed in FOXML format outside the repository, then ingested in a single step.

For a conceptual overview of these and other special Fedora object types, please see the Fedora Digital Object Model document.

On this page:

Persistent Identifier

As with all Fedora objects, Service Deployments must declare a persistent identifier ("PID") that is unique within the repository. There are no special restrictions on the PID beyond those described in the Fedora Identifiers document. In FOXML format, the PID is declared in the root element of the XML document as shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<foxml:digitalObject xmlns:foxml="info:fedora/fedora-system:def/foxml#"
    VERSION="1.1" PID="demo:MyServiceDeployment">
  <!-- Object Properties -->
  <!-- Datastreams -->
</foxml:digitalObject>
```

Object Properties

There are also no special restrictions on the top-level object properties for Service Deployments. In the following FOXML fragment, we assert that the state of the object is *Active* and the label is *My Service Deployment*.

```
<foxml:objectProperties>
  <foxml:property
      NAME="info:fedora/fedora-system:def/model#state"
      VALUE="Active"/>
  <foxml:property
      NAME="info:fedora/fedora-system:def/model#label"
      VALUE="My Service Deployment"/>
</foxml:objectProperties>
```

DC Datastream

The DC datastream describes the object in human terms using the Dublin Core Metadata Element Set. If you don't provide it yourself, Fedora will automatically add a bare-bones DC datastream for the object at ingest time. For an example DC datastream, please see the FOXML Ingest Example.

RELS-EXT Datastream

The RELS-EXT datastream expresses relationships and properties of the object in RDF. An object asserts that it is a Service Deployment Object through a special *hasModel* relationship to the *fedora-system:ServiceDeployment-3.0* object. SDeps must also have two additional relationships asserted in RELS-EXT:

- an *isDeploymentOf* relationship to the associated Service Definition. This tells Fedora which set of methods are deployed by this SDep.
- an *isContractorOf* relationship to the associated Content Model. This tells Fedora which set of objects this SDep is capable of operating on.

```
<foxml:datastream ID="RELS-EXT"
    CONTROL_GROUP="X" STATE="A" VERSIONABLE="true">
  <foxml:datastreamVersion ID="RELS-EXT1.0"
      MIMETYPE="application/rdf+xml"
      FORMAT_URI="info:fedora/fedora-system:FedoraRELSExt-1.0"
      LABEL="RDF Statements about this object">
    <foxml:xmlContent>
      <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:fedora-model="info:fedora/fedora-system:def/model#">
        <rdf:Description rdf:about="info:fedora/demo:MyServiceDeployment">
          <fedora-model:hasModel
              rdf:resource="info:fedora/fedora-system:ServiceDeployment-3.0"/>
          <fedora-model:isDeploymentOf
              rdf:resource="info:fedora/demo:MyServiceDefinition"/>
          <fedora-model:isContractorOf
              rdf:resource="info:fedora/demo:MyContentModel"/>
        </rdf:Description>
      </rdf:RDF>
    </foxml:xmlContent>
  </foxml:datastreamVersion>
</foxml:datastream>
```

METHODMAP Datastream

The METHODMAP datastream in the Service Deployment object is similar to the one stored in the corresponding Service Definition, but it also indicates which datastreams (if any) are required as input to each deployed method. This information is specified in the *Service Deployment* Method Map format, which is an extension of the *Service Definition* Method Map format.

When creating a Service Deployment method map, it is helpful to copy the method map from the corresponding Service Definition, then add information as necessary.

Continuing with the example method map set out in Creating a Service Definition, below we will specify that:

- *methodOne* accepts a single datastream as input, whose ID is "FOO".
- *methodTwo* accepts no datastreams as input.
    - it includes the user input parameter "param1" specified in the SDef
    - it includes a default input parameter "uri" with default value $objuri, allowing the URI of the Fedora object to be used as a parameter when calling an external service
- *methodThree* accepts three datastreams as input, whose IDs are "FOO", "BAR", and "BAZ".
    - it also includes the user input parameters specified in the SDef

- it also includes a default input parameter "pid" with default value $pid, allowing the PID of the Fedora object to be used as a parameter when calling an external service

Default input parameters

Default input parameters can be specified in the method definitions. These are parameters that are then available to the WSDL's specification of the external service in a similar way to user input parameters and datastreams. The default value of the parameter can either be specified as some constant value, or the following "special" variables may be used which will be substituted when the method is invoked:

- `$pid` - will be substitued with the (URL-encoded) value of the Fedora object's PID, eg `changeme:1234`
- `$objuri` - will be substituted with the (URL-encoded) value of the Fedora object's URI, eg `info:fedora/changeme:1234`

Note: Differences from the METHODMAP specified in the corresponding Service Definition are indicated below with a preceding "+" sign.

```
  <foxml:datastream ID="METHODMAP"
        CONTROL_GROUP="X" STATE="A" VERSIONABLE="true">
    <foxml:datastreamVersion ID="METHODMAP1.0"
+       FORMAT_URI="info:fedora/fedora-system:FedoraSDepMethodMap-1.1"
+       LABEL="Deployment Method Map" MIMETYPE="text/xml">
      <foxml:xmlContent>
        <fmm:MethodMap
              xmlns:fmm="http://fedora.comm.nsdlib.org/service/methodmap"
              name="N/A">
          <fmm:Method operationName="methodOne"
+             wsdlMsgName="methodOneRequest"
+             wsdlMsgOutput="response">
+           <fmm:DatastreamInputParm parmName="FOO"
+              passBy="URL_REF" required="true"/>
+           <fmm:MethodReturnType wsdlMsgName="response"
+              wsdlMsgTOMIME="N/A"/>
          </fmm:Method>
          <fmm:Method operationName="methodTwo"
+             wsdlMsgName="methodTwoRequest"
+             wsdlMsgOutput="response">
            <fmm:UserInputParm parmName="parm1" defaultValue="value1"
              required="false" passBy="VALUE"/>
+           <fmm:DefaultInputParm parmName="uri" defaultValue="$objuri"
+              passBy="VALUE" required="true"/>
+           <fmm:MethodReturnType wsdlMsgName="response"
+              wsdlMsgTOMIME="N/A"/>
          </fmm:Method>
          <fmm:Method operationName="methodThree"
+             wsdlMsgName="methodThreeRequest"
+             wsdlMsgOutput="response">
            <fmm:UserInputParm parmName="parm1" defaultValue="value1"
              required="false" passBy="VALUE">
              <fmm:ValidParmValues>
                <fmm:ValidParm value="value1"/>
                <fmm:ValidParm value="value2"/>
              </fmm:ValidParmValues>
            </fmm:UserInputParm>
            <fmm:UserInputParm parmName="parm2" defaultValue=""
              required="true" passBy="VALUE"/>
+           <fmm:DefaultInputParm parmName="pid" defaultValue="$pid"
+              passBy="VALUE" required="true"/>
+           <fmm:DatastreamInputParm parmName="FOO"
+              passBy="URL_REF" required="true"/>
+           <fmm:DatastreamInputParm parmName="BAR"
+              passBy="URL_REF" required="true"/>
+           <fmm:DatastreamInputParm parmName="BAZ"
+              passBy="URL_REF" required="true"/>
+           <fmm:MethodReturnType wsdlMsgName="response"
+              wsdlMsgTOMIME="N/A"/>
          </fmm:Method>
        </fmm:MethodMap>
      </foxml:xmlContent>
    </foxml:datastreamVersion>
  <foxml:datastream>
```

DSINPUTSPEC Datastream

The DSINPUTSPEC datastream specifies additional information about each datastream. It is written in the Fedora Datastream Input Specification format.

```
<foxml:datastream ID="DSINPUTSPEC"
      CONTROL_GROUP="X" STATE="A" VERSIONABLE="true">
  <foxml:datastreamVersion ID="DSINPUTSPEC1.0"
      MIMETYPE="text/xml"
      FORMAT_URI="info:fedora/fedora-system:FedoraDSInputSpec-1.1"
      Label="Datastream Input Specification">
    <foxml:xmlContent>
      <fbs:DSInputSpec
          xmlns:fbs="http://fedora.comm.nsdlib.org/service/bindspec"
          label="N/A">
        <fbs:DSInput wsdlMsgPartName="FOO"
            DSMax="1" DSMin="1" DSOrdinality="false">
          <fbs:DSInputLabel/>N/A</fbs:DSInputLabel>
          <fbs:DSMIME>N/A</fbs:DSMIME>
          <fbs:DSInputInstruction>N/A</fbs:DSInputInstruction>
        </fbs:DSInput>
        <fbs:DSInput wsdlMsgPartName="BAR"
              DSMax="1" DSMin="1" DSOrdinality="false">
          <fbs:DSInputLabel/>N/A</fbs:DSInputLabel>
          <fbs:DSMIME>N/A</fbs:DSMIME>
          <fbs:DSInputInstruction>N/A</fbs:DSInputInstruction>
        </fbs:DSInput>
        <fbs:DSInput wsdlMsgPartName="BAZ" pid="demo:MyContentModel"
              DSMax="1" DSMin="1" DSOrdinality="false">
          <fbs:DSInputLabel/>N/A</fbs:DSInputLabel>
          <fbs:DSMIME>N/A</fbs:DSMIME>
          <fbs:DSInputInstruction>N/A</fbs:DSInputInstruction>
        </fbs:DSInput>
      </fbs:DSInputSpec>
    </foxml:xmlContent>
  </foxml:datastreamVersion>
</foxml:datastream>
```

Notice that the BAZ datastream input specification above has an extra attribute, "pid" specified. When present, the pid attribute tells Fedora that the datastream to be used as input is located inside a specific object. In this case, it is the BAZ datastream in the "demo:MyContentModel" object. Normally, each input datastream is expected to come from the particular data object from which the method is invoked. When a pid is specified, the associated datastream effectively serves as a constant.

WSDL Datastream

The WSDL datastream brings everything together, describing to Fedora exactly what needs to be done under the hood when a dissemination request is made. For each method, the WSDL provides enough information to Fedora so that it may do a URL replacement in order to fulfill a request.

Fedora does not work with arbitrary WSDL in this datastream; it must be in a very specific form. Notably, Fedora currently only supports performing disseminations via HTTP GET. You may use the following as a template for your own WSDL datastreams, changing the messages, port type, and binding sections as appropriate.

To customize this for your own use:

- Leave the "response" wsdl:message as-is, but replace the others with your own. You should have a wsdl:message element for each method defined by the Service Definition. By convention, the name of each message should be the method name followed by the word "Request" (e.g. "methodOneRequest"). Within each wsdl:message element, add a wsdl:part for each user and/or datastream input parameter, and specify all types as "this:inputType".
- Within the wsdl:portType section, replace each existing operation with your own. Each operation should reference the corresponding request method as the input, and point to "this:response" as the output.
- Finally, in the wsdl:binding section, replace each existing operation with your own. In this section, the location attribute is the important bit. The location should given as a URL template, where all occurances of (VARIABLE) are automatically replaced with the appropriate value at runtime. If VARIABLE denotes a datastream, it will be replaced with a URL reference to the datastream content at runtime. If VARIABLE denotes a user input parameter, it will replaced with the URL-encoded value of the variable at runtime. If VARIABLE is "PID" and your METHODMAP specifies $PID as the default value for the PID variable as in the example METHODMAP above , it will be replaced with the data object's URL-encoded PID at runtime.

```
<foxml:datastream ID="WSDL"
    CONTROL_GROUP="X" STATE="A" VERSIONABLE="true">
```

```
<foxml:datastreamVersion ID="WSDL1.0"
    MIMETYPE="text/xml"
    FORMAT_URI="http://schemas.xmlsoap.org/wsdl/"
    LABEL="WSDL Bindings">
  <foxml:xmlContent>
    <wsdl:definitions name="definitions"
        targetNamespace="urn:thisNamespace"
        xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
        xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap"
        xmlns:soapenc="http://schemas.xmlsoap.org/wsdl/soap/encoding"
        xmlns:this="urn:thisNamespace"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <wsdl:types>
        <xsd:schema targetNamespace="urn:thisNamespace">
          <xsd:simpleType name="inputType">
            <xsd:restriction base="xsd:string"/>
          </xsd:simpleType>
        </xsd:schema>
      </wsdl:types>
      <wsdl:message name="methodOneRequest">
        <wsdl:part name="FOO" type="this:inputType"/>
      </wsdl:message>
      <wsdl:message name="methodTwoRequest">
        <wsdl:part name="parm1" type="this:inputType"/>
        <wsdl:part name="uri" type="this:inputType"/>
      </wsdl:message>
      <wsdl:message name="methodThreeRequest">
        <wsdl:part name="parm1" type="this:inputType"/>
        <wsdl:part name="parm2" type="this:inputType"/>
        <wsdl:part name="pid" type="this:inputType"/>
        <wsdl:part name="FOO" type="this:inputType"/>
        <wsdl:part name="BAR" type="this:inputType"/>
        <wsdl:part name="BAZ" type="this:inputType"/>
      </wsdl:message>
      <wsdl:message name="response">
        <wsdl:part name="response" type="xsd:base64Binary"/>
      </wsdl:message>
      <wsdl:portType name="portType">
        <wsdl:operation name="methodOne">
          <wsdl:input message="this:methodOneRequest"/>
          <wsdl:output message="this:response"/>
        </wsdl:operation>
        <wsdl:operation name="methodTwo">
          <wsdl:input message="this:methodTwoRequest"/>
          <wsdl:output message="this:response"/>
        </wsdl:operation>
        <wsdl:operation name="methodThree">
          <wsdl:input message="this:methodThreeRequest"/>
          <wsdl:output message="this:response"/>
        </wsdl:operation>
      </wsdl:portType>
      <wsdl:service name="service">
        <wsdl:port binding="this:binding" name="port">
          <http:address location="LOCAL"/>
        </wsdl:port>
      </wsdl:service>
      <wsdl:binding name="binding" type="this:portType">
        <http:binding verb="GET"/>
        <wsdl:operation name="methodOne">
          <http:operation location="(FOO)"/>
          <wsdl:input>
            <http:urlReplacement/>
          </wsdl:input>
          <wsdl:output>
            <mime:content type="N/A"/>
          </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="methodTwo">
          <http:operation
```

```
                location="http://local.fedora.server/fedora/risearch?format=(parm1)&amp;type=triples&amp;
lang=spo&amp;query=(uri)+*+*"/>
              <wsdl:input>
                <http:urlReplacement/>
              </wsdl:input>
              <wsdl:output>
                <mime:content type="N/A"/>
              </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="methodThree">
              <http:operation
                  location="http://example.org/service?a=(parm1)&amp;b=(parm2)&amp;c=(parm3)&amp;d=(FOO)&amp;e=
(BAR)&amp;f=(BAZ)&amp;g=(pid)"/>
              <wsdl:input>
                <http:urlReplacement/>
              </wsdl:input>
              <wsdl:output>
                <mime:content type="N/A"/>
              </wsdl:output>
            </wsdl:operation>
          </wsdl:binding>
        </wsdl:definitions>
      </foxml:xmlContent>
    </foxml:datastreamVersion>
  </foxml:datastream>
```