Soap Attachments using WSIF

1 Overview

WSIF supports passing attachments in a Mime message using the Axis provider. The attachment is a javax.activation.DataHandler, The mime:multipartRelated, mime:part and mime:content tags are used to describe the attachment in the WSDL.

The WSDL extensions...

```
<binding name="MyBinding" type="tns:abc" >
 <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="MyOperation">
  <soap:operation soapAction=""/>
  <input>
   <mime:multipartRelated>
    <mime:part>
     <soap:body use="encoded" namespace="http://mynamespace"
       encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </mime:part>
    <mime:part>
     <mime:content part="attch" type="text/html"/>
    </mime:part>
   </mime:multipartRelated>
  </input>
 </operation>
</binding>
```

The above WSDL demonstrates a simple operation that has one attachment called *attch*. There must be a part called *attch* on the input message for MyOperation. There may be other input parts to MyOperation that are not attachments. In the binding input there must either be a <soap:body or a <mime:multipartRelated, but not both. For mime messages, the soap:body is inside a mime:part. There must only be one mime:part that contains a soap:body in the binding input and that must not contain a mime:content as well, since a content type of text/xml is assumed for the soap:body. There can be multiple attachments in a mime message, each described by a mime:part. Each mime:part (that isn't a soap:body) contains a mime:content that describes the attachment itself. The type field inside the mime:content is not checked or used by WSIF. Instead it is there to provide a hint to the application using WSIF as to what the attachment is intended to contain. Multiple mime:contents inside a single mime:part means that the backend service will expect a single attachment with a type specified by one of the mime:contents inside that mime:part. The parts="..." tag (optional)

inside the soap:body is assumed to contain the names of all the mime parts as well as the names of all the soap parts in the message.

2 Passing attachments to WSIF

The following code snippet could invoke the service described by the WSDL above...

```
import javax.activation.DataHandler;
. . .
DataHandler dh = new DataHandler(new FileDataSource("axis.jpg"));
WSIFServiceFactory factory = WSIFServiceFactory.newInstance();
WSIFService service =
factory.getService("my.wsdl",null,null,"http://mynamespace","abc");
WSIFOperation op = service.getPort().createOperation("MyOperation");
WSIFMessage in = op.createInputMessage();
in.setObjectPart("attch",dh);
op.executeInputOnlyOperation(in);
```

I use tomcat with soap 2.3 as my soap server so my DeploymentDescriptor.xml contains the following type mapping..

```
<isd:mappings>
<isd:map encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:x="http://mynamespace"
 qname="x:datahandler"
 javaType="javax.activation.DataHandler"
 java2XMLClassName="org.apache.soap.encoding.soapenc.MimePartSerializer"
 xml2JavaClassName="org.apache.soap.encoding.soapenc.MimePartSerializer" />
</isd:mappings>
```

and my backend service gets invoked with the following signature ...

```
public void MyOperation(DataHandler dh);
```

Attachments can also be passed in to WSIF using stubs...

```
DataHandler dh = new DataHandler(new FileDataSource("axis.jpg"));
WSIFServiceFactory factory = WSIFServiceFactory.newInstance();
WSIFService service =
factory.getService("my.wsdl",null,null,"http://mynamespace","abc");
MyInterface stub = (MyInterface)service.getStub(MyInterface.class);
stub.MyOperation(dh);
```

Attachments can also be returned from an operation, but at present only one attachment can be returned as the return parameter.

3 Types and type mappings

By default, attachments are passed into WSIF as DataHandlers. If the part on the message which is the DataHandler maps to a mime:part in the WSDL, then WSIF will automatically

map the fully qualified name of the WSDL type to DataHandler.class and set up that type mapping with Axis.

In your WSDL you may have defined a schema for the attachment as a binary[] for instance. Whether or not you have done this, WSIF silently ignores this mapping and treats the attachment as a DataHandler, unless you have explicitly issued mapType(). WSIF lets axis set the mime content type based on the DataHandler's type, instead of the mime:content type specified in the WSDL.

4 Restrictions

These are not supported...

- Attachments using the Apache Soap provider
- Mime/Axis/Jms
- DIME
- Passing in axis AttachmentParts
- Passing in javax.xml.transform.Source and javax.mail.internet.MimeMultipart
- The mime:mimeXml WSDL tag
- Attachments over doc-style
- Nesting a mime:multipartRelated inside a mime:part
- Types that extend DataHandler, Image, etc
- Types that contain DataHandler, Image, etc
- Arrays or Vectors of DataHandlers, Images, etc
- Multiple inout or output attachments