Web services programming tips and tricks: 
Attachments using the wsi:swaRef XML type from WSI

Russell Butek
June 27, 2006

This tip describes how to use the XML attachment type defined by the Web services Interoperability organization: wsi:swaRef.

Introduction

IBM developerWorks offers articles with various approaches to sending Web services attachments (see the Related topics section for links). This article addresses yet another mechanism, the Web Service Interoperability (WS-I) organization's SOAP with Attachments reference type: wsi:swaRef.

Why wsi:swaRef?

WS-I created an XML type for attachments since none existed. The WSDL specification contains a definition for attachments, but it falls short for a number of reasons:

- Attachments are not visible in the interface
- Only message parts can contain attachments
- Attachments and document/literal wrapped don't mix well

Attachments are not visible in the interface

Attachments are defined in SOAP, and the WSDL specification maps SOAP attachments to the WSDL binding. This means that the interface portion of a WSDL document -- portType, messages, types -- is not aware of attachments.

For example, take a look at Listing 1. This listing is from the article "Web services programming tips and tricks: SOAP attachments with JAX-RPC" (see the Related topics section.) If you ignore the binding for a moment and only look at the WSDL's interface, you'll see two operations, sendImage and sendOctet. Each of these has a hexBinary input. JAX-RPC maps hexBinary to a Java byte[]. Using only this information, you could reasonably guess that this WSDL interface would map to a Java interface which would have two methods, each of which would have a single byte[] parameter. This is a good guess, and for anything but MIME types, you'd be correct. But for MIME types you must inspect the binding to determine the real parameter types. In the binding you'll see that the inputs for the operations are really MIME content types -- image/jpeg and application/octet-stream -- instead of hexBinary. JAX-RPC maps these to java.awt.Image and javax.activation.DataHandler, respectively.
For attachments, the WSDL interface is not sufficient to determine the Java interface. This is unfortunate. The line between interface and implementation is blurred.

**Listing 1. WSDL with attachment before WS-I's wsi:swaRef**

```xml
<definitions
    xmlns="http://schemas.xmlsoap.org/wsdl/
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/
    targetNamespace="urn:attachment.tip"
    xmlns:tns="urn:attachment.tip">
    <message name="empty"/>
    <message name="imageMsg">
        <part name="image" type="xsd:hexBinary"/>
    </message>
    <message name="octetMsg">
        <part name="octet" type="xsd:hexBinary"/>
    </message>
    <portType name="AttachmentTip">
        <operation name="sendImage">
            <input message="tns:imageMsg"/>
            <output message="tns:empty"/>
        </operation>
        <operation name="sendOctet">
            <input message="tns:octetMsg"/>
            <output message="tns:empty"/>
        </operation>
    </portType>
</definitions>
```
Only message parts can contain attachments

Notice, in Listing 1, that the binding correlates a MIME type to a message part. This means that all attachments must be parts; you cannot define an attachment to be an element of a complexType, for instance. This is very restrictive and minimizes an attachment's usefulness.

Attachments and the document/literal wrapped pattern

The third problem is that attachments do not fit well into the current industry's inclination towards the document/literal wrapped pattern of WSDL. The article, "Which style of WSDL should I use?", linked to in the Related topics section, describes this wrapped pattern. Some people believe attachments do not fit into the document/literal wrapped pattern at all. I would argue otherwise, but regardless of what I think, there is no specification which has attempted to make them fit.

The problem is that, in the document/literal wrapped pattern, there can be only one part in a message and that part must refer to a wrapper element. How does the attachment part fit into this model? It does not. It could fit if you relax the definition of what can be in a message, but no one has attempted to formalize this relaxation.

WS-I's solution

By creating an XML attachment type, WS-I has alleviated all of these problems. A wsi:swaRef is visible in a WSDL 'interface'. A wsi:swaRef's location is not limited to a message part, it could be an element of a complexType. As an XML type, wsi:swaRef fits nicely into the document/literal wrapped pattern, which requires parameters to be elements of a complexType. listing 2 is a modification of Listing 1 with standard attachments replaced by wsi:swaRefs. Note that we have also turned Listing 2's WSDL into a document/literal wrapped WSDL.

Listing 2. WSDL with WS-I's wsi:swaRef attachment type

```xml
<definitions
    xmlns="http://schemas.xmlsoap.org/wsdl/
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/
    targetNamespace="urn:attachment.tip"
    xmlns:tns="urn:attachment.tip">
  <types>
    <schema
        targetNamespace="urn:attachment.tip"
        xmlns:tns="urn:attachment.tip"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:wsi="http://ws-i.org/profiles/basic/1.1/xsd">
      <import
        namespace="http://ws-i.org/profiles/basic/1.1/xsd"
        schemaLocation="http://ws-i.org/profiles/basic/1.1/xsd"/>
      <element name="sendImage">
        <complexType>
            ...
        </complexType>
      </element>
    </schema>
  </types>
</definitions>
```
<sequence>
  <element name="image" type="wsi:swaRef"/>
</sequence>
</complexType>
</element>
<element name="sendImageResponse">
  <complexType>
    <sequence/>
  </complexType>
</element>
<element name="sendOctet">
  <complexType>
    <sequence>
      <element name="octet" type="wsi:swaRef"/>
    </sequence>
  </complexType>
</element>
<element name="sendOctetResponse">
  <complexType>
    <sequence/>
  </complexType>
</element>
</schema>
</types>
<message name="imageMsg">
  <part name="parameters" element="tns:sendImage"/>
</message>
<message name="imageResponseMsg">
  <part name="return" element="tns:sendImageResponse"/>
</message>
<message name="octetMsg">
  <part name="parameters" element="tns:sendOctet"/>
</message>
<message name="octetResponseMsg">
  <part name="return" element="tns:sendOctetResponse"/>
</message>
<portType name="AttachmentTip">
  <operation name="sendImage">
    <input message="tns:imageMsg"/>
    <output message="tns:imageResponseMsg"/>
  </operation>
  <operation name="sendOctet">
    <input message="tns:octetMsg"/>
    <output message="tns:octetResponseMsg"/>
  </operation>
</portType>
(binding name="AttachmentBinding" type="tns:AttachmentTip">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="sendImage">
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="sendOctet">
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
You can see in Listing 2 that wsi:swaRef is very simple to define. It behaves just like any other primitive XML type. However, in my opinion, it is too simple. Unlike the standard attachment, you do not know the MIME type of the attachment. All you know is that you have an attachment. It would have been more useful if WS-I had defined wsi:swaRef such that it had an attribute, for instance, which provided the MIME type, but WS-I is not in the business of defining specifications, just clarifying them, so when they do actually define something new, they prefer to keep it to a minimum.

**wsi:swaRef and SOAP**

Listing 2 showed you how wsi:swaRef looks in a WSDL definition. **Listing 3** shows you a sample SOAP request message. It is the message for the sendImage operation of the service defined in the WSDL in Listing 2. You can see from the highlighted items how the image wsi:swaRef parameter correlates to content ID of the actual image data. While not exactly the same as a normal SOAP with attachment message, the concept is similar. The article "Tip: Passing files to a Web service" in the the **Related topics** section offers an example of a SOAP with attachment message.

**Listing 3. SOAP message with wsi:swaRef data**

```xml
<soapenv:Envelope
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header/>
   <soapenv:Body>
     <a985:sendImage xmlns:a985="urn:attachment.tip">
       <image>cid:image=881207433652.1147796619984.IBM.WEBSERVICES@ISSW</image>
     </a985:sendImage>
   </soapenv:Body>
</soapenv:Envelope>
```

... binary image data ...
**wsi:swaRef and Java**

You've seen the declaration of wsi:swaRef; and you've seen an instance of wsi:swaRef in a SOAP message. The last thing to consider is how wsi:swaRef maps to a programming language. Since I'm a Java aficionado, I'll discuss the Java mapping.

WS-I defined wsi:swaRef after the JAX-RPC version 1.1 specification was finalized, so JAX-RPC does not provide us with a mapping. However, it does give us a hint. It states that unknown MIME types are mapped to `javax.activation.DataHandler`. Since wsi:swaRef is really an unknown MIME type, it seems reasonable that it also maps to `javax.activation.DataHandler`. And if we look forward a little, the new JAX-B 2.0 specification does, in fact, dictate this mapping. So it is a pretty safe guess that any implementation which supports wsi:swaRef will map it to `javax.activation.DataHandler`. **Listing 4** shows the Java interface for the WSDL in Listing 2.

**Listing 4. Java mapping of wsi:swaRef**

```java
public interface AttachmentTip extends java.rmi.Remote {
    public void sendImage(javax.activation.DataHandler image)
        throws java.rmi.RemoteException;
    public void sendOctet(javax.activation.DataHandler octet)
        throws java.rmi.RemoteException;
}
```

**Summary**

WSDL's attachment definition has some problems, so the WS-I organization created the new attachment type wsi:swaRef. This maps into the SOAP message in a similar manner as a standard WSDL attachment does; and this maps to Java as an unknown MIME type. That is, `javax.activation.DataHandler`.
Related topics

- "Tip: Passing files to a Web service" (developerWorks, February, 2004) discusses the different solutions available for passing binary data (typically files) to a Web service.
- "Web services programming tips and tricks: SOAP attachments with JAX-RPC" (developerWorks, February, 2004) describes how you can use JAX-RPC APIs to send MIME attachments.
- To understand the document/literal wrapped pattern, read "Which style of WSDL should I use?" (developerWorks, May, 2005). This article describes the WSDL and SOAP messages for each combination to help you decide how which combination of style and use to use.
- Read W3C's SOAP with Attachments specification.
- Java API for XML-Based RPC (JAX-RPC) Downloads & Specifications provides links to the JAX-RPC 1.1 specification.
- Read the near-final draft of the Java Architecture for XML Binding (JAXB) 2.0 specification for the XML-to-Java mapping of the future.
- Find the WS-I's Basic Profile and Attachment Profile on their Deliverables page.
- Download a free trial version of WebSphere Application Server Version 6.1.
- Build your next development project with IBM trial software, available for download directly from developerWorks.