MTOM

Using MTOM

SOAP Message Transmission Optimization Mechanism (MTOM) specifies an optimized method for sending binary data as part of a SOAP message. Unlike SOAP with Attachments, MTOM requires the use of XML-binary Optimized Packaging (XOP) packages for transmitting binary data. Using MTOM to send binary data does not require you to fully define the MIME Multipart/Related message as part of the SOAP binding. It does, however, require that you do the following:

1. Annotate the data that you are going to send as an attachment.
   You can annotate either your WSDL or the Java class that implements your data.
2. Enable the runtime's MTOM support.
   This can be done either programatically or through configuration.
3. Develop a `DataHandler` for the data being passed as an attachment.

Annotating Data Types to use MTOM

Overview

When defining a data type for passing along a block of binary data, such as an image file or a sound file, in WSDL you define the element for the data to be of type `xsd:base64Binary`. By default, any element of type `xsd:base64Binary` results in the generation of a `byte[]` which can be serialized using MTOM. However, the default behavior of the code generators does not take full advantage of the serialization.

In order to fully take advantage of MTOM you must add annotations to either your service's WSDL document or the JAXB class that implements the binary data structure. Adding the annotations to the WSDL document forces the code generators to generate streaming data handlers for the binary data. Annotating the JAXB class involves specifying the proper content types and may also involve changing the type specification of the field containing the binary data.

WSDL first

The following example shows a WSDL document for a Web service that uses a message which contains one string field, one integer field, and a binary field. The binary field is intended to carry a large image file, so it is not appropriate for sending along as part of a normal SOAP message.

Message for MTOM

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="XrayStorage"
    targetNamespace="http://mediStor.org/x-rays"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://mediStor.org/x-rays"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:xsd1="http://mediStor.org/types/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <types>
    <schema targetNamespace="http://mediStor.org/types/"
            xmlns="http://www.w3.org/2001/XMLSchema">
      <complexType name="xRayType">
        <sequence>
          <element name="patientName" type="xsd:string" />
          <element name="patientNumber" type="xsd:int" />
          <element name="imageData" type="xsd:base64Binary" />
        </sequence>
      </complexType>
      <element name="xRay" type="xsd1:xRayType" />
    </schema>
  </types>

  <message name="storRequest">
    <part name="record" element="xsd1:xRay"/>
  </message>
  <message name="storResponse">
    <part name="success" type="xsd:boolean"/>
  </message>

  <portType name="xRayStorage">
    <operation name="store">
      <input message="tns:storRequest" name="storRequest"/>
      <output message="tns:storResponse" name="storResponse"/>
    </operation>
  </portType>

  <binding name="xRayStorageSOAPBinding" type="tns:xRayStorage">
    <soap12:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="store">
      <soap12:operation soapAction="" style="document"/>
      <input name="storRequest">
        <soap12:body use="literal"/>
      </input>
      <output name="storResponse">
        <soap12:body use="literal"/>
      </output>
    </operation>
  </binding>
  ...
</definitions>
```

If you wanted to use MTOM to send the binary part of the message as an optimized attachment you would need to add the `xmime:expectedContentTypes` attribute to the element containing the binary data. This attribute is defined in the `http://www.w3.org/2005/05/xmlmime` namespace and specifies the MIME types that the element is expected to contain. You can specify a comma separated list of MIME types. The setting of this attribute will change how the code generators create the JAXB class for the data. For most MIME types, the code generator will create a `DataHandler`. Some MIME types, such as those for images, have defined mappings.

The MIME types are maintained by the Internet Assigned Numbers Authority (IANA) and described in detail in **Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies** (ftp://ftp.isi.edu/in-notes/rfc2045.txt) and **Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types**(ftp://ftp.isi.edu/in-notes/rfc2046.txt)

For most uses you would specify `application/octet-stream`.

The following example shows how you would modify `xRayType` to use MTOM.

Binary Data for MTOM

```
...
  <types>
    <schema targetNamespace="http://mediStor.org/types/"
            xmlns="http://www.w3.org/2001/XMLSchema"
            xmlns:xmime="http://www.w3.org/2005/05/xmlmime">
      <complexType name="xRayType">
        <sequence>
          <element name="patientName" type="xsd:string" />
          <element name="patientNumber" type="xsd:int" />
          <element name="imageData" type="xsd:base64Binary"
                   xmime:expectedContentTypes="application/octet-stream"/>
        </sequence>
      </complexType>
      <element name="xRay" type="xsd1:xRayType" />
    </schema>
  </types>
...
```

The generated JAXB class generated for `xRayType` will no longer contain a `byte[]`. Instead the code generator will see the `xmime:expectedContentTypes` attribute and generate a `DataHandler` for the `imageData` field.

You do not need to change the binding element to use MTOM. The runtime will make the appropriate changes when the data is sent.

Java first

If you are doing Java first development you can make your JAXB class MTOM ready by doing the following:

1. Make sure the field holding the binary data is a `DataHandler`.
2. Add the `@XmlMimeType()` annotation to the field containing the data you want to be streamed as an MTOM attachment.

The following example shows a JAXB class annotated for using MTOM.

JAXB Class for MTOM

```
@XmlType
public class XRayType {
    protected String patientName;
    protected int patientNumber;
    @XmlMimeType("application/octet-stream")
    protected DataHandler imageData;
  ...
}
```

Enabling MTOM

By default the runtime does not enable MTOM support. It will send all binary data as either part of the normal SOAP message or as an unoptimized attachment. You can activate MTOM support either programatically or through the use of configuration.

Using JAX-WS APIs

Both clients and servers need to have the MTOM optimizations enabled. The JAX-WS APIs offer different mechanisms for each type of endpoint.

server

If you published your server using the JAX-WS APIs you enable the runtime's MTOM support as follows:

1. Get access to the `Endpoint` object for your published service.
   The easiest way to get the `Endpoint` object is when you publish the endpoint.
2. Get the SOAP binding from the `Endpoint` using its `getBinding()` method as shown below.

Getting the SOAP Binding from an Endpoint

```
// Endpoint ep is declared previously
SOAPBinding binding = (SOAPBinding)ep.getBinding();
```

You must cast the returned binding object to a `SOAPBinding` object in order to access the MTOM property.

3. Set the bindings MTOM enabled property to true using the binding's setMTOMEnabled() method as shown below.

Setting a Service Provider's MTOM Enabled Property

```
binding.setMTOMEnabled(true);
```

Client

To MTOM enable a JAX-WS client you do the following:

1. Cast the client's proxy to a `BindingProvider` object.
2. Get the SOAP binding from the `BindingProvider` using its `getBinding()` method as shown below.

Getting a SOAP Binding from a BindingProvider

```
// BindingProvider bp declared previously
SOAPBinding binding = (SOAPBinding)bp.getBinding();
```

3. Set the bindings MTOM enabled property to `true` using the binding's `setMTOMEnabled()` method as shown below.

Setting a Consumer's MTOM Enabled Property

```
binding.setMTOMEnabled(true);
```

Using configuration

Overview

If you publish your service using XML, such as when deploying into a container, you can enable your endpoint's MTOM support in the endpoint's configuration file.

Procedure

The MTOM property is set inside the `jaxws:endpoint` element for your endpoint. To enable MTOM do the following:

1. Add a `jaxws:property` child element to the endpoint's `jaxws:endpoint` element.
2. Add a `entry` child element to the `jaxws:property` element.
3. Set the `entry` element's `key` attribute to `mtom-enabled`.
4. Set the `entry` element's `value` attribute to `true`.

Example

The following example shows an endpoint that is MTOM enabled.

Configuration for Enabling MTOM

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:jaxws="http://cxf.apache.org/jaxws"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
                           http://cxf.apache.org/jaxws http://cxf.apache.org/schema/jaxws.xsd">

  <jaxws:endpoint id="xRayStorage"
                  implementor="demo.spring.xRayStorImpl"
                  address="http://localhost/xRayStorage">
    <jaxws:properties>
      <entry key="mtom-enabled" value="true"/>
    </jaxws:properties>
  </jaxws:endpoint>
</beans>
```