Tip: Implement implicit and explicit SOAP headers

Learn the differences and how to create your own SOAP headers

Andre Tost February 15, 2005

You can define SOAP headers in a WSDL definition using what are commonly called explicit and implicit headers. Learn the difference between these two styles and how these differences might impact you when developing with JAX-RPC.

View more content in this series

The SOAP specification describes that a SOAP envelope can contain an optional header part. This header is meant to carry data that is not part of the payload of the actual message. The WSDL specification defines how to declare SOAP header data as part of a Web services definition. There are two ways to define SOAP headers in a WSDL definition: *explicit* and *implicit* headers.

SOAP header styles

One typical use for SOAP headers is to transfer contextual data. For example, if a message includes a digital signature, this signature will most likely be transferred in the SOAP header. Another example is for Web services that support some form of session with a client. Once such a session is established, they require that a specific identifier be sent along with every request. The WS-AtomicTransaction specification (see Related topics) also describes a very similar mechanism for running a coordinated sequence of interactions among several Web services.

The WSDL specification provides you with two different ways of specifying the use of SOAP header fields. In *explicit headers*, you add all of the header information to the portType of the service. It becomes exposed to the client as additional parameters. The advantage of this style is that the client can pass all of the information directly to the service. The downside of this style is that it often clutters the external interface of a service with information that is not related to its business purpose at all.

This is exactly where using *implicit headers* can help: header information is not part of the portType, and thus does not impact the functional interface of the service. On the other hand, implicit headers are harder to deal with programmatically.

Before diving into more detail on the programming side of things, let's look at how these different styles are defined.

SOAP header binding types in WSDL

The easiest way to describe the different styles of SOAP headers is to begin with an example. The following WSDL extract in Listing 1 is taken from a previous article that explained the use of SOAP headers:

Listing 1. SOAP header bindings in WSDL

```
<wsdl:definitions targetNamespace="http://soapheader.ibm.com" ...>
 <wsdl:types ...>
  <schema elementFormDefault="qualified" ...>
 ...
      <element name="quote_timestamp" type="xsd:dateTime" />
  </schema>
 </wsdl:types>
<wsdl:binding name="StockServiceSoapBinding" type="intf:StockService">
      <wsdlsoap:binding style="document" transport=
      "http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="getLastSellPrice">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getLastSellPriceRequest">
            <wsdlsoap:header message="intf:getLastSellPriceRequest" part=
            "request_header" use="literal"/>
            <wsdlsoap:body parts="parameters" use="literal"/>
         </wsdl:input>
         <wsdl:output name="getLastSellPriceResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
...
</wsdl:definitions>
```

You can see that at a particular place within the binding section of the WSDL file an element called `<wsdlsoap:header>` is used. It is contained in the `<wsdl:input>` element, which tells you that there is a SOAP header section as part of the request message for an operation. The content of the `<wsdlsoap:header>` element identifies the message part that is to be transferred in the header.

That's pretty straightforward, but is this an explicit or implicit header? Well, from the extract above, you can't exactly tell. It could be either way. And this is why: the header binding defines that the part named *request_header* of the message `intf:getLastSellPriceRequest` goes into the header part of the SOAP envelope. The header style depends on whether this message part is used in the portType of the Web service or not. Let's look at both cases in detail.

Explicit headers

You can call a header definition *explicit* if it is part of the service `<portType>`. In other words, the message part named *request_header* must be used somewhere in the portType, as shown in Listing 2:

Listing 2. Explicit SOAP header in WSDL

```
<wsdl:message name="getLastSellPriceRequest">
      <wsdl:part element="intf:getLastSellPrice" name="parameters"/>
      <wsdl:part name="request_header" element="intf:quote_timestamp"/>
   </wsdl:message>

   <wsdl:message name="getLastSellPriceResponse">
      <wsdl:part element="intf:getLastSellPriceResponse" name="parameters"/>
   </wsdl:message>

   <wsdl:portType name="StockService">
      <wsdl:operation name="getLastSellPrice">
         <wsdl:input message="intf:getLastSellPriceRequest" name=
         "getLastSellPriceRequest"/>
         <wsdl:output message="intf:getLastSellPriceResponse" name=
         "getLastSellPriceResponse"/>
      </wsdl:operation>
   </wsdl:portType>
```

Note that the message named `getLastSellPriceRequest` has two parts in it. One part is going into the body part of the SOAP request message, the other one into the header. Listing 3 shows the relevant part of the WSDL file that shows the two parts:

Listing 3. Explicit SOAP header in WSDL - SOAP bindings

```
<wsdl:input name="getLastSellPriceRequest">
      <wsdlsoap:header message="intf:getLastSellPriceRequest" part=
      "request_header" use="literal"/>
      <wsdlsoap:body parts="parameters" use="literal"/>
</wsdl:input>
```

The `<portType>` element defines the external interface of a Web service. It defines which data is sent as part of a request message. If some of the request data is transferred in the SOAP header part of the request message, you can call this an explicit header. The same holds true for the case where part (or all) of the response message is defined as a header element, respectively.

Implicit headers

This case will be simple, right? If the message part that is transferred in the header does not show up anywhere in the `<portType>` element, you have an *implicit* header. Listing 4 shows what this might look like:

Listing 4. Implicit SOAP header in WSDL

```
<wsdl:message name="getLastSellPriceRequest">
      <wsdl:part element="intf:getLastSellPrice" name="parameters"/>
   </wsdl:message>

   <wsdl:message name="getLastSellPriceResponse">
      <wsdl:part element="intf:getLastSellPriceResponse" name="parameters"/>
   </wsdl:message>

   <wsdl:message name="getLastSellPriceRequestHeader">
      <wsdl:part name="request_header" element="intf:quote_timestamp"/>
   </wsdl:message>

   <wsdl:portType name="StockService">
      <wsdl:operation name="getLastSellPrice">
         <wsdl:input message="intf:getLastSellPriceRequest" name=
         "getLastSellPriceRequest"/>
         <wsdl:output message="intf:getLastSellPriceResponse" name=
         "getLastSellPriceResponse"/>
      </wsdl:operation>
   </wsdl:portType>
```

The listing shows three messages defined, and only two of them are used in the `<portType>`. The third one, called `getLastSellPriceRequestHeader`, is not used at all. Now assume that the SOAP binding looks what Listing 5 shows:

Listing 5. Implicit SOAP header in WSDL - SOAP Bindings

```
<wsdl:input name="getLastSellPriceRequest">
      <wsdlsoap:header message="intf:getLastSellPriceRequestHeader" part=
      "request_header" use="literal"/>
      <wsdlsoap:body parts="parameters" use="literal"/>
</wsdl:input>
```

> There might be cases where a SOAP header is sent as part of a message, even though the header is not declared in the WSDL definition at all. You could consider this another form of an implicit header. Generally, however, it is considered a bad style to add parts to a SOAP message that are not contained in its WSDL definition, so I won't cover this case in this paper.

Again, the `<wsdlsoap:header>` element refers to a message part that is not used in the `<portType>`. Hence, you have an implicit header.

SOAP message represention

Note that the wire format of each type of SOAP header is the same: information is carried in the SOAP header part of the message. In other words, you can't tell when looking at a SOAP message whether it was defined as an implicit or explicit header in the associated WSDL definition. Listing 6 shows the SOAP message for a sample Web service:

Listing 6. SOAP message with header

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" ...>
 <soapenv:Header>
      <p677:quote_timestamp xmlns:p677=
      "http://soapheader.ibm.com">2005-01-03T06:00:00.000Z</p677:quote_timestamp>
 </soapenv:Header>
 <soapenv:Body>
  <p677:getLastSellPrice xmlns:p677="http://soapheader.ibm.com">
   <p677:ticker>IBM</p677:ticker>
  </p677:getLastSellPrice>
 </soapenv:Body>
</soapenv:Envelope>
```

JAX-RPC mapping of explicit and implicit headers

Now that you have a definition of both styles of headers, what does this mean if you're developing Web services with JAX-RPC? The JAX-RPC specification talks about something called implicit and explicit *service context*. Even though a service context might or might not be mapped into SOAP header definitions, you can safely assume here that they mean the same thing. The specification then describes how explicit service contexts are mapped into the remote interface of a service, whereas implicit contexts are typically not.

Practically speaking, this means that explicit headers (or service contexts) are relatively easy to handle because they are mapped to the *Service Endpoint Interface* (SEI) of the Web service. The message part that is transferred in the SOAP header becomes an additional parameter of the SEI. For example, the SEI for the WSDL with the explicit definition above looks like what Listing 7 shows:

Listing 7. Service Endpoint Interface for explicit header type

```
public interface StockServiceExplicit_PortType extends java.rmi.Remote {
    public com.ibm.soapheader.GetLastSellPriceResponse
      getLastSellPrice(com.ibm.soapheader.GetLastSellPrice parameters,
          java.util.Calendar request_header)
              throws java.rmi.RemoteException;
}
```

At run-time, the value passed in as the `request_header` parameter is put into the SOAP header of the request message.

You can probably guess by now what the interface will look like in the implicit header case. There is no reference to the header message part at all! The tooling simply ignores it and generates the SEI shown in Listing 8:

Listing 8. Service Endpoint Interface for implicit header type

```
public interface StockServiceImplicit_PortType extends java.rmi.Remote {
 public float getLastSellPrice(java.lang.String ticker) throws java.rmi.RemoteException;
}
```

While I won't go further into this in this paper, it's interesting to note that, in this case, the input and output parameters are not wrapped into a new class as before.

Implicit header handling in Java programming

So how do you deal with a Web service that defines an implicit header in JAX-RPC? The specification does not help you there, because only those elements that are included in the portType are generated into the Service Endpoint Interface.

Beyond that, different JAX-RPC vendors might offer different levels of support. One way of dealing with implicit headers is to use JAX-RPC handlers to manage the content of the header. Handlers have full access to the SOAP message on both the client and server side, including the header. But how can you pass the right content from the client application into the handler? You can use the `_setProperty()` on the `javax.xml.rpc.Stub` interface to pass information to the client proxy object. From there, the handler can retrieve it using the `javax.xml.rpc.handler.SOAPMessageContext.getProperty()` method. The sample that comes with this paper shows you this mechanism (click the **code** icon at the top or bottom of this paper to download the EAR file).

On the server side, you can also use a JAX-RPC handler to deal with implicit headers. Moreover, you can pass SOAP header information into a service implementation through the `javax.xml.rpc.server.ServiceLifecycle` interface. I explained how to use this interface in a previous tip in this series. And again, refer to the sample code for a complete example.

Summary

There are two different ways to define the use of SOAP header fields in a Web service, namely implicit and explicit headers. The difference between the two is easily detectable in the WSDL definition. While it might seem to be a minor design detail, it has a significant impact on the generated JAX-RPC Java code. Especially in the case of implicit headers, extra code must be written (or generated) to deal with the header information that is not part of the portType.

Downloadable resources

Description	Name	Size
Source code (EAR file)	ws-tip-soap-headerstyles.ear	83 KB

Related topics

- Browse all of the *Web services programming tips and tricks* articles on developerWorks.
- Find a brief introduction to SOAP headers in the tip Using SOAP headers with JAX-RPC (developerWorks, October 2003).