SOAP 1.1

Adding a SOAP 1.1 Binding

Using wsdltosoap

To generate a SOAP 1.1 binding using **wsdltosoap** use the following command:

```
wsdl2soap [[-?] | [-help] | [-h]] {-iport-type-name} [-bbinding-name] [-doutput-directory] [-ooutput-file]
[-nsoap-body-namespace] [-style (document/rpc)] [-use (literal/encoded)] [-v ] [[ -verbose ] |  [ -quiet ]]
wsdlurl
```

The command has the following options:

Option	Interpretation
-?	Displays the online help for this utility.
-help	
-h	
-iport-type-name	Specifies the portType element for which a binding should be generated.
-bbinding-name	Specifies the name of the generated SOAP binding.
-doutput-directory	Specifies the directory to place generated WSDL file.
-ooutput-file	Specifies the name of the generated WSDL file.
-nsoap-body-namespace	Specifies the SOAP body namespace when the style is RPC.
-style (document/rpc)	Specifies the encoding style (document or RPC) to use in the SOAP binding. The default is document.
-use (literal/encoded)	Specifies the binding use (encoded or literal) to use in the SOAP binding. The default is literal.
-v	Displays the version number for the tool.
-verbose	Displays comments during the code generation process.
-quiet	Suppresses comments during the code generation process.
wsdlurl	The path and name of the WSDL file containing the portType element definition.

The `-i port-type-name` and `wsdlurl` arguments are required. If the `-style rpc` argument is specified, the `-n soap-body-namspace` argument is also required. All other arguments are optional and may be listed in any order.

wsdltosoap does not support the generation of document/encoded SOAP bindings.

Example

If your system had an interface that took orders and offered a single operation to process the orders it would be defined in a WSDL document similar to the one shown below.

Ordering System Interface

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="widgetOrderForm.wsdl"
    targetNamespace="http://widgetVendor.com/widgetOrderForm"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://widgetVendor.com/widgetOrderForm"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd1="http://widgetVendor.com/types/widgetTypes"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">

<message name="widgetOrder">
  <part name="numOrdered" type="xsd:int"/>
</message>
<message name="widgetOrderBill">
  <part name="price" type="xsd:float"/>
</message>
<message name="badSize">
  <part name="numInventory" type="xsd:int"/>
</message>

<portType name="orderWidgets">
  <operation name="placeWidgetOrder">
    <input message="tns:widgetOrder" name="order"/>
    <output message="tns:widgetOrderBill" name="bill"/>
    <fault message="tns:badSize" name="sizeFault"/>
  </operation>
</portType>
...
</definitions>
```

The SOAP binding generated for orderWidgets is shown below.

Binding for orderWidgets

```
<binding name="orderWidgetsBinding" type="tns:orderWidgets">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="placeWidgetOrder">
      <soap:operation soapAction="" style="document"/>
      <input name="order">
        <soap:body use="literal"/>
      </input>
      <output name="bill">
        <soap:body use="literal"/>
      </output>
      <fault name="sizeFault">
        <soap:body use="literal"/>
      </fault>
  </operation>
</binding>
```

This binding specifies that messages are sent using the document/literal message style.

Adding SOAP Headers to a SOAP 1.1 Binding

Overview

SOAP headers are defined by adding `soap:header` elements to your default SOAP 1.1 binding. The `soap:header` element is an optional child of the `input`, `output`, and `fault` elements of the binding. The SOAP header becomes part of the parent message. A SOAP header is defined by specifying a message and a message part. Each SOAP header can only contain one message part, but you can insert as many SOAP headers as needed.

Syntax

The syntax for defining a SOAP header is shown in **SOAP Header Syntax**. The `message` attribute of `soap:header` is the qualified name of the message from which the part being inserted into the header is taken. The `part` attribute is the name of the message part inserted into the SOAP header. Because SOAP headers are always document style, the WSDL message part inserted into the SOAP header must be defined using an element. Together the message and the part attributes fully describe the data to insert into the SOAP header.

SOAP Header Syntax

```
<binding name="headwig">
  <soap:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="weave">
      <soap:operation soapAction="" style="document"/>
      <input name="grain">
        <soap:body .../>
        <soap:header message="QName" part="partName"/>
      </input>
...
</binding>
```

As well as the mandatory `message` and `part` attributes, `soap:header` also supports the `namespace`, the `use`, and the `encodingStyle` attributes. These optional attributes function the same for `soap:header` as they do for `soap:body`.

Splitting messages between body and header

The message part inserted into the SOAP header can be any valid message part from the contract. It can even be a part from the parent message which is being used as the SOAP body. Because it is unlikely that you would want to send information twice in the same message, the SOAP binding provides a means for specifying the message parts that are inserted into the SOAP body.

The `soap:body` element has an optional attribute, `parts`, that takes a space delimited list of part names. When `parts` is defined, only the message parts listed are inserted into the SOAP body. You can then insert the remaining parts into the SOAP header.

When you define a SOAP header using parts of the parent message, CXF automatically fills in the SOAP headers for you.

Example

SOAP 1.1 Binding with a SOAP Header shows a modified version of the orderWidgets service shown in **Ordering System Interface**. This version has been modified so that each order has an `xsd:base64binary` value placed in the SOAP header of the request and response. The SOAP header is defined as being the `keyVal` part from the `widgetKey` message. In this case you would be responsible for adding the SOAP header in your application logic because it is not part of the input or output message.

SOAP 1.1 Binding with a SOAP Header

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="widgetOrderForm.wsdl"
    targetNamespace="http://widgetVendor.com/widgetOrderForm"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://widgetVendor.com/widgetOrderForm"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd1="http://widgetVendor.com/types/widgetTypes"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">

<types>
  <schema targetNamespace="http://widgetVendor.com/types/widgetTypes"
          xmlns="http://www.w3.org/2001/XMLSchema"
          xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <element name="keyElem" type="xsd:base64Binary"/>
  </schema>
</types>

<message name="widgetOrder">
  <part name="numOrdered" type="xsd:int"/>
</message>
<message name="widgetOrderBill">
  <part name="price" type="xsd:float"/>
</message>
<message name="badSize">
  <part name="numInventory" type="xsd:int"/>
</message>
<message name="widgetKey">
  <part name="keyVal" element="xsd1:keyElem"/>
</message>

<portType name="orderWidgets">
  <operation name="placeWidgetOrder">
    <input message="tns:widgetOrder" name="order"/>
    <output message="tns:widgetOrderBill" name="bill"/>
    <fault message="tns:badSize" name="sizeFault"/>
  </operation>
</portType>

<binding name="orderWidgetsBinding" type="tns:orderWidgets">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="placeWidgetOrder">
      <soap:operation soapAction="" style="document"/>
      <input name="order">
        <soap:body use="literal"/>
        <soap:header message="tns:widgetKey" part="keyVal"/>
      </input>
      <output name="bill">
        <soap:body use="literal"/>
        <soap:header message="tns:widgetKey" part="keyVal"/>
      </output>
      <fault name="sizeFault">
        <soap:body use="literal"/>
      </fault>
  </operation>
</binding>
...
</definitions>
```

You could modify **SOAP 1.1 Binding with a SOAP Header** so that the header value was a part of the input and output messages as shown in **SOAP 1.1 Binding for orderWidgets with a SOAP Header**. In this case `keyVal` is a part of the input and output messages. In the `soap:body` element's parts attribute specifies that `keyVal` is not to be inserted into the body. However, it is inserted into the SOAP header.

SOAP 1.1 Binding for orderWidgets with a SOAP Header

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="widgetOrderForm.wsdl"
    targetNamespace="http://widgetVendor.com/widgetOrderForm"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://widgetVendor.com/widgetOrderForm"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd1="http://widgetVendor.com/types/widgetTypes"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">

<types>
  <schema targetNamespace="http://widgetVendor.com/types/widgetTypes"
          xmlns="http://www.w3.org/2001/XMLSchema"
          xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <element name="keyElem" type="xsd:base64Binary"/>
  </schema>
</types>

<message name="widgetOrder">
  <part name="numOrdered" type="xsd:int"/>
  <part name="keyVal" element="xsd1:keyElem"/>
</message>
<message name="widgetOrderBill">
  <part name="price" type="xsd:float"/>
  <part name="keyVal" element="xsd1:keyElem"/>
</message>
<message name="badSize">
  <part name="numInventory" type="xsd:int"/>
</message>

<portType name="orderWidgets">
  <operation name="placeWidgetOrder">
    <input message="tns:widgetOrder" name="order"/>
    <output message="tns:widgetOrderBill" name="bill"/>
    <fault message="tns:badSize" name="sizeFault"/>
  </operation>
</portType>

<binding name="orderWidgetsBinding" type="tns:orderWidgets">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="placeWidgetOrder">
      <soap:operation soapAction="" style="document"/>
      <input name="order">
        <soap:body use="literal" parts="numOrdered"/>
        <soap:header message="tns:widgetOrder" part="keyVal"/>
      </input>
      <output name="bill">
        <soap:body use="literal" parts="bill"/>
        <soap:header message="tns:widgetOrderBill" part="keyVal"/>
      </output>
      <fault name="sizeFault">
        <soap:body use="literal"/>
      </fault>
  </operation>
</binding>
...
</definitions>
```