ID2208 Programming Web Services

Service description WSDL

Mihhail Matskin:
http://people.kth.se/~misha/ID2208/index

Spring 2016

Content

WSDL

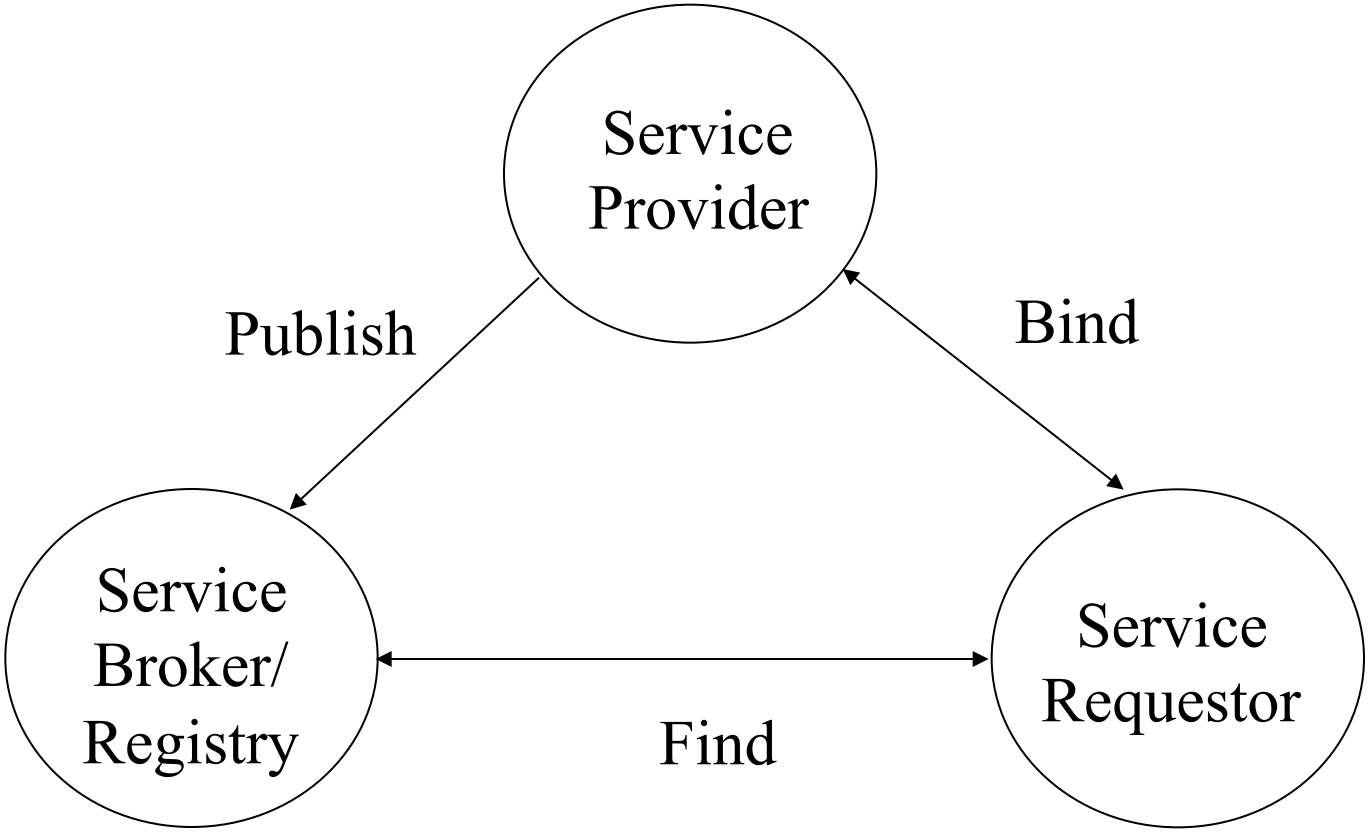
- Introduction
- What should service describe
- Web service description components
- Bindings
- Non-functional description
- WS-Policy
- WSDL 2.0 vs. WSDL 1.1

This lecture reference

Text-book **Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, 2nd Edition**

Chapter 4

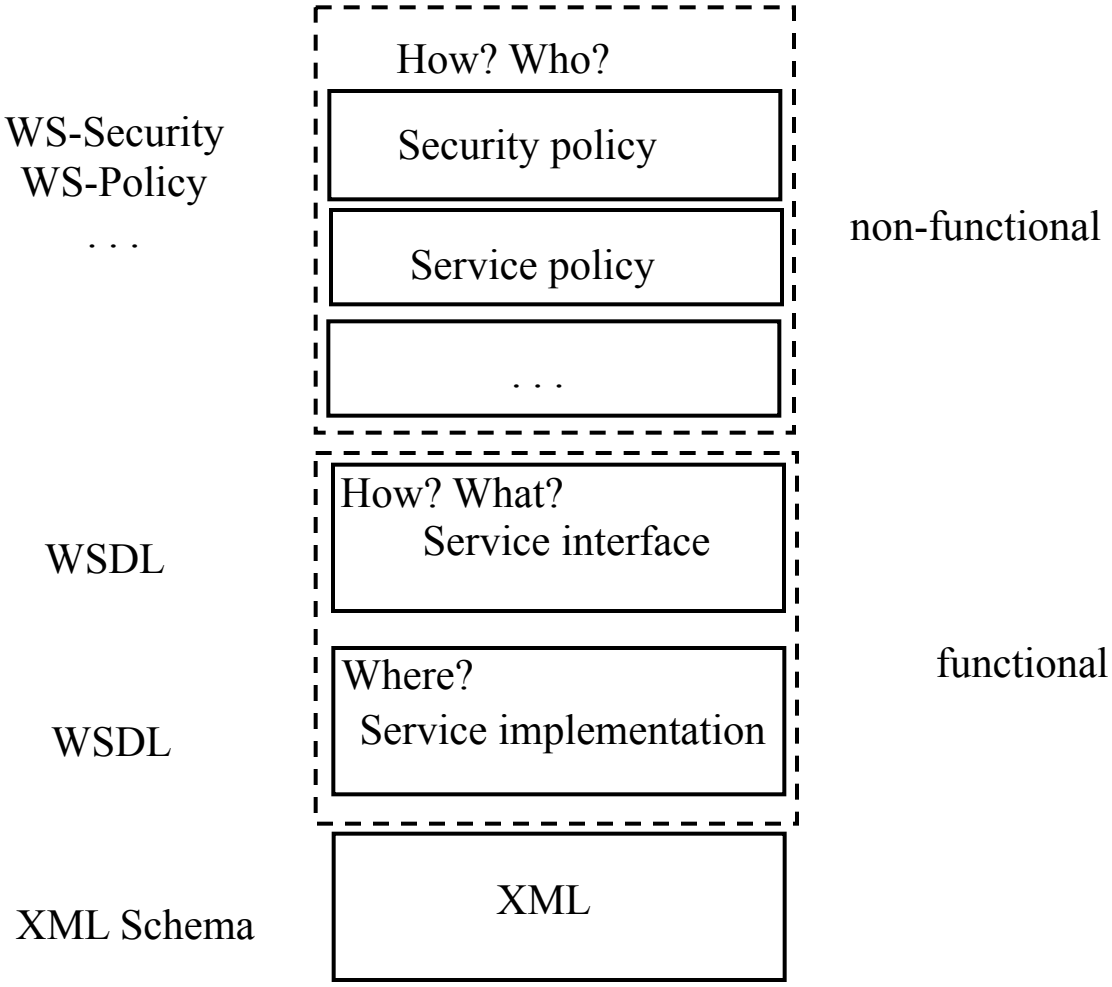
Service Oriented Architecture (individual roles)



Service Description

- How to know what kind of messages to send to a service?
- What XML to put into the body?
- What format the response message might come in?
- Where the response to be expected?
- What message protocol to use?
- What network address to send message to?

What Should Service Describe?



WSDL Essentials

- What service does – operations
- How a service is accessible – details of data format and protocols necessary to access service's operations
- Where a service is located – details of protocol specific network addresses (URL)

WSDL Essentials

- Interface information describing all publicly available functions
- Data type information for all message requests and message responses
- Binding information about the transport protocol to be used
- Address information for locating the specified service

The WSDL Specification

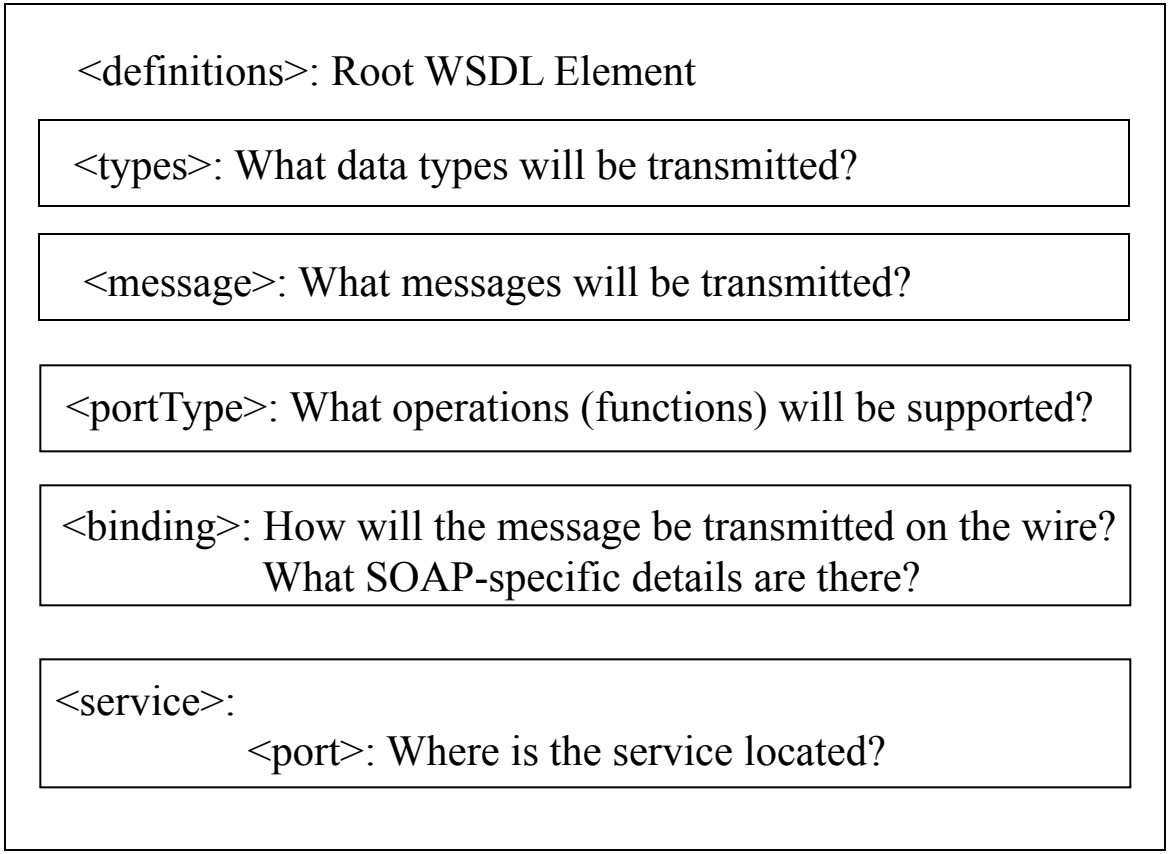
- **definitions**
- **types**
- **message**
- **portType**

(**interface** in WSDL 2.0)

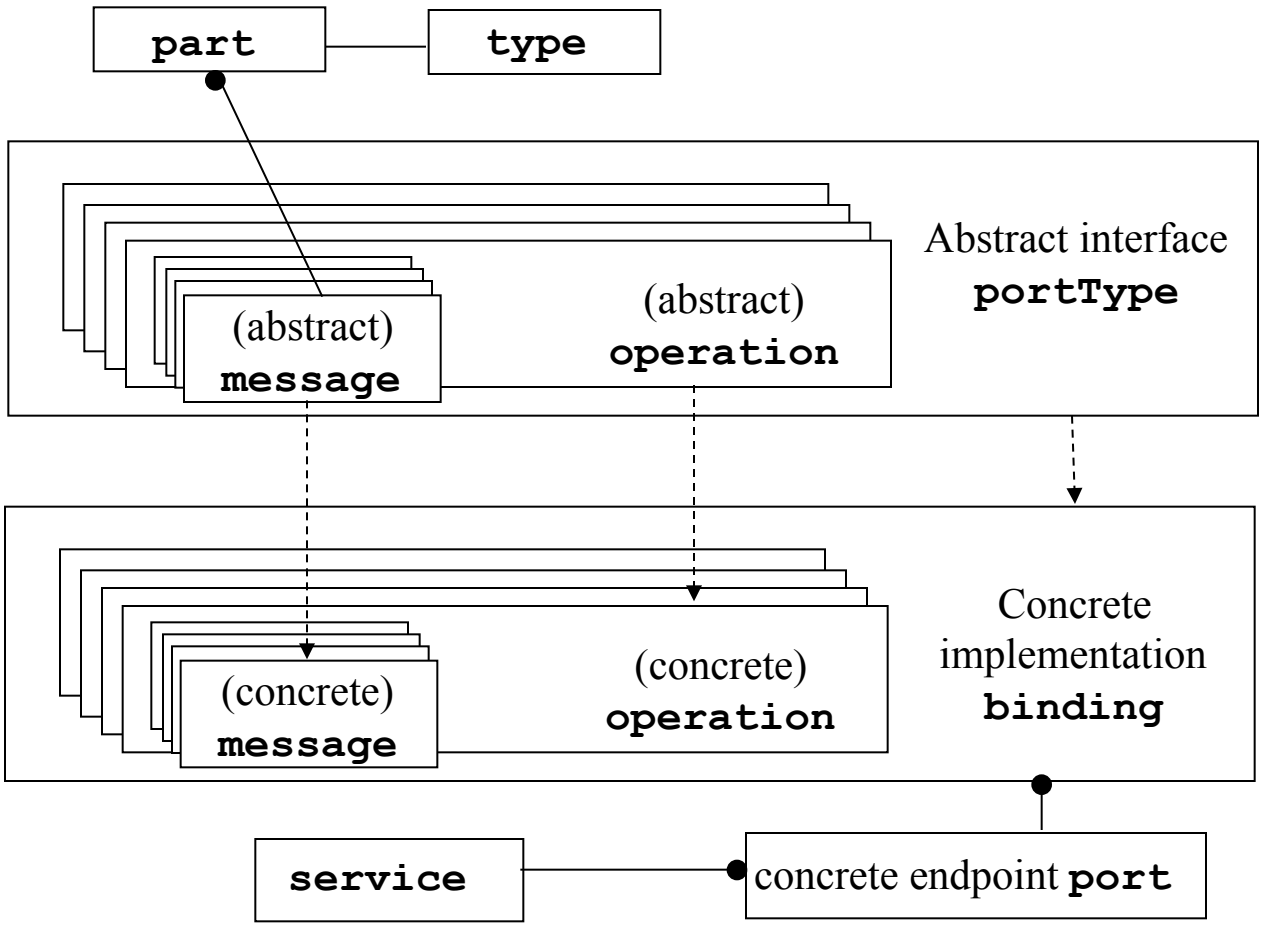
- **binding**
- **port**

(**endPoint** in WSDL 2.0)

- **service**



WSDL Information Model



Additional elements

- **`documentation`**
  - The documentation element is used to provide human-readable documentation and can be included inside any other WSDL element.

- **`import`**
  - The import element is used to import other WSDL documents or XML Schemas. This enables more modular WSDL documents.

HelloService.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService"
     targetNamespace="http://www.ecerami.com/wsdl/HelloService"
     xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:tns="http://www.ecerami.com/wsdl/HelloService" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <message name="SayHelloRequest">
            <part name="firstName" type="xsd:string"/>
     </message>
     <message name="SayHelloResponse">
            <part name="greeting" type="xsd:string"/>
     </message>
     <portType name="Hello_PortType">
            <operation name="sayHello">
                 <input message="tns:SayHelloRequest"/>
                 <output message="tns:SayHelloResponse"/>
            </operation>
     </portType>
     <binding name="Hello_Binding" type="tns:Hello_PortType">
            <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="sayHello">
                 <soap:operation soapAction="sayHello"/>
                 <input>
                        <soap:body
                             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                             namespace="urn:examples:helloservice" use="encoded"/>
                 </input>
                 <output>
                        <soap:body
                             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/“
                             namespace="urn:examples:helloservice" use="encoded"/>
                 </output>
            </operation>
     </binding>
     <service name="Hello_Service">
            <documentation> WSDL File for HelloService</documentation>
            <port binding="tns:Hello_Binding" name="Hello_Port">
                 <soap:address location="http://localhost:8080/soap/servlet/rpcrouter"/>
            </port>
     </service>
</definitions>
```

HelloService.wsdl

<definitions>: The Hello Service

<message>:
1) sayHelloRequest: firstName parameter
2) sayHelloResponse: greeting return value

<portType>: sayHello operation that consists of a request/ response service

<binding>: Direction to use the SOAP HTTP transport protocol

<service>: Service available at:
http://localhost:8080/soap/servlet/rpcrouter

`definitions`

- The root element of WSDL document
- Contains all other WSDL elements
  - **`documentation`**
  - **`import`**
  - **`types`**
  - **`message`**
  - **`portType`**
  - **`binding`**
  - **`service`**

definitions

```
<definitions name="HelloService"
   targetNamespace=
       "http://www.ecerami.com/wsdl/HelloService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns=
       "http://www.ecerami.com/wsdl/HelloService"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
…
</definitions>
```

`message`

- A collection of parts
- Each message element can be used as an input, output or fault message
- All message elements must have a unique name in the WSDL document

message

```
<message name="SayHelloRequest">
  <part name="firstName" type="xsd:string"/>
</message>
```

```
<message name="PriceCheckResponse">
      <part name="result"
            element="avail:StockAvailability"/>
</message>
```

message

```
<message
  name="StockAvailableExpirationNotification">
      <part name="timeStamp"
            type="xsd:timeInstant"/>
      <part name="correlationID“
            type="reg:correlationID"/>
      <part name="items"
            type="reg:ArrayOfItem"/>
      <part name="clientArg"
            type="xsd:string"/>
</message>
```

`portType`

- Describes an interface to a Web service
- WSDL document can contain zero or more `portType` elements
- Typically a WSDL document contains a single `portType`

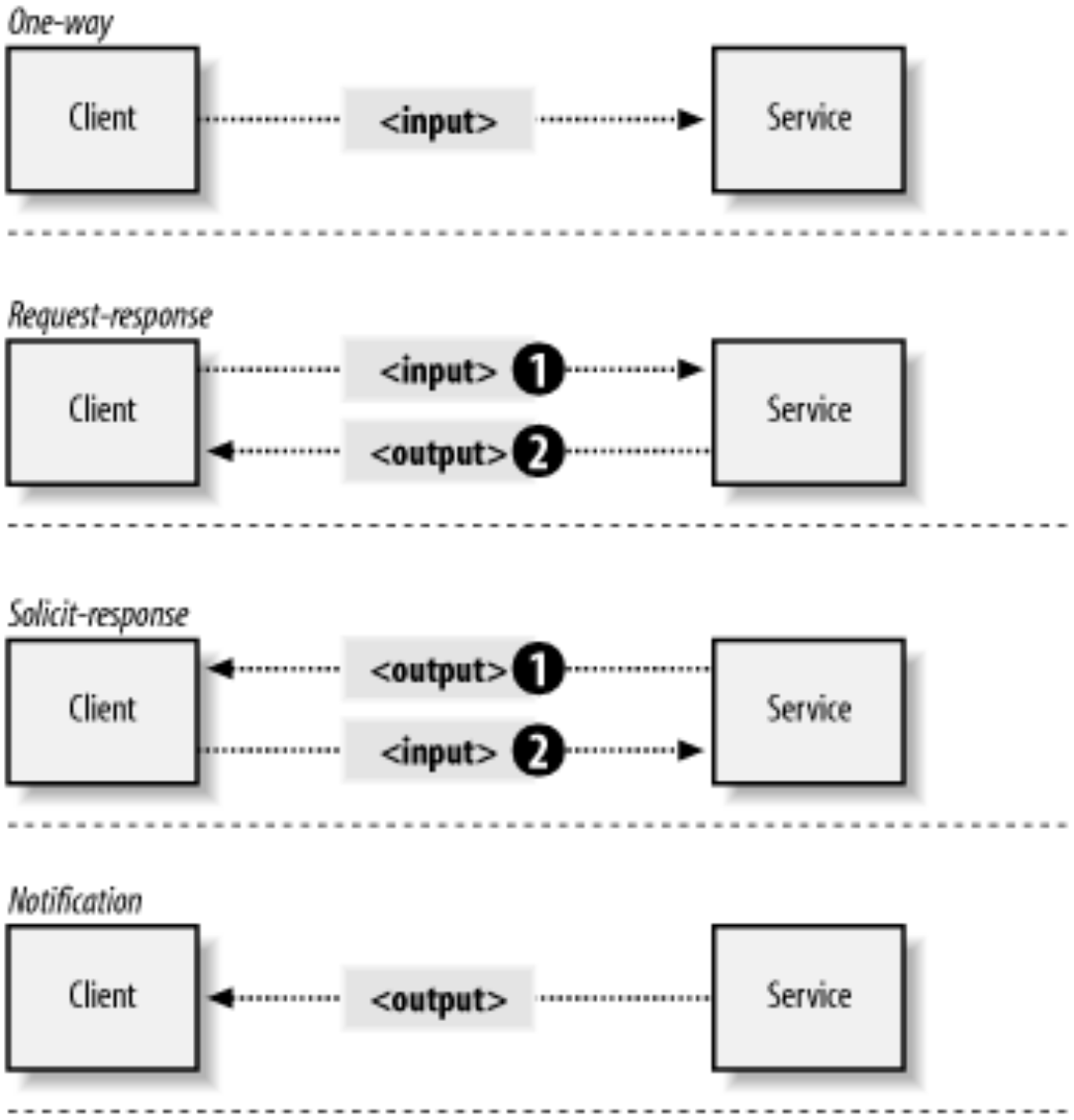
`operations`

- Equivalent to a method signature in Java
- Defines a method on a Web service including its name and parameters
- Names should be different with the names of **`portTypes`**

portType and operations

```
<portType name="Hello_PortType">
  <operation name="sayHello">
     <input message="tns:SayHelloRequest"/>
     <output message="tns:SayHelloResponse"/>
  </operation>
</portType>
```

Operation patterns



operations

```
<operation name="registration">
  <input  message="StockAvailableRegistrationRequest"/>
  <output message="StockAvailableRegistrationResponse"/>
  <fault  message="StockAvailableRegistrationError"
          name="StockAvailableNotificationErrorMessage"/>
  <fault  message="StockAvailableExpirationError"
          name="StockAvailableExpirationError"/>
</operation>
```

`binding`

- Bindings can be made available via multiple transports, including HTTP GET, HTTP POST, or SOAP
- The binding element itself specifies name and type attributes:

```
<binding name="Hello_Binding"
         type="tns:Hello_PortType">
```

- The type attribute references the **`portType`** defined earlier in the document

SOAP binding

- **`soap:binding`**
  - This element indicates that the binding will be made available via SOAP.
  - A style attribute indicates the overall style of the SOAP message format
  - The transport attribute indicates the transport of the SOAP messages.
- **`soap:operation`**
  - This element indicates the binding of a specific operation to a specific SOAP implementation.
- **`soap:body`**
  - This element enables you to specify the details of the input and output messages.

WSDL Example (priceCheck)

```
<message name="PriceCheckRequest">
  <part name="sku" element="avail:sku"/>
</message>
<message name="PriceCheckResponse">
  <part name="result"
        element="avail:stockAvailability">
</message>

<portType name="PriceCheckPortType">
  <operation name="checkPrice">
      <input message="pc:PriceCheckRequest"/>
      <output message="pc:PriceCheckResponse"/>
  </operation>
</portType>
```

SOAP binding

```
<binding name="PriceCheckSOAPBinding"
         type="pc:PriceCheckPortType">
   <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="checkPrice">
        <soap:operation soapAction=
             "http://www.skatestown.com/services/PriceCheck/>
        <input>
             <soap:body use="literal" />
        </input>
        <output>
             <soap:body use="literal" />
        </output>
   </operation>
</binding>
```

Input message appearance

```
<input>
     <soap:body use="literal" />
</input>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
   soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema
   xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
     <sku xmlns="http://www.skatestown.com/ns/availability">
               123
     </sku>
   </soapenv:body>
</soapenv:Envelope>
```

Service response

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <SOAP-ENV:Body>
        <StockAvailability
            xmlns="http://www.skatetown.com/ns/availability">
              <sku>123</sku>
              <price>100.0</price>
              <quantityAvailable>12</quantityAvailable>
        </StockAvailability>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP binding

```
<binding name="Hello_Binding" type="tns:Hello_PortType">
 <soap:binding style="rpc"
       transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="sayHello">
  <soap:operation soapAction="sayHello"/>
  <input>
   <soap:body
       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
       namespace="http://www.ecerami.com/wsdl/HelloService"
       use="encoded"/>
  </input>
  <output>
   <soap:body encodingStyle=
       "http://schemas.xmlsoap.org/soap/encoding/"
       namespace="http://www.ecerami.com/wsdl/HelloService"
       use="encoded"/>
  </output>
 </operation>
</binding>
```

Input message appearance

```
<input>
 <soap:body
      encodingStyle="http://schemas.xmlsoap.org/soap/
      encoding/"
      namespace="http://www.ecerami.com/wsdl/HelloService"
      use="encoded"/>
</input>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv=
      "http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema
      xmlns:xsi="http://www.w3.org/2001/XMLSchema- instance">
  <soapenv:Body>
     <sh1:SayHello xmlns:sh1="http://www.ecerami.com/wsdl/
           HelloService" soapenv:encodingStyle="http://
           schemas.xmlsoap.org/soap/encoding/">
              <arg1 xsi:type="xsd:string">World</ar1>
      </sh1:SayHello>
  </soapenv:body>
</soapenv:Envelope>
```

`service`

- Specifies location of a service (contains a set of `port` elements)
- Includes a documentation element to provide human-readable documentation

service and port

```
<service name="Hello_Service">
  <documentation>
     WSDL File for HelloService
  </documentation>
  <port binding="Hello_Binding"
        name="Hello_Port">
    <soap:address   location=
   "http://localhost:8080/soap/servlet/rcrouter">
  </port>
</service>
```

XMethods eBay Price Watcher Service

```
<?xml version = "1.0"?>
<definitions name = "eBayWatcherService" xmlns = http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd = "http://www.w3.org/1999/XMLSchema"
   targetNamespace = "http://www.xmethods.net/sd/eBayWatcherService.wsdl"
    xmlns:tns = "http://www.xmethods.net/sd/eBayWatcherService.wsdl"
    xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/">
 <message name = "getCurrentPriceRequest">
    <part name = "auction_id" type = "xsd:string"/>
 </message>
 <message name = "getCurrentPriceResponse">
    <part name = "return" type = "xsd:float"/>
 </message>
 <portType name = "eBayWatcherPortType">
    <operation name = "getCurrentPrice">
     <input  message = "getCurrentPriceRequest"  name = "getCurrentPrice"/>
     <output message = "getCurrentPriceResponse" name = "getCurrentPriceResponse"/>
   </operation>
 </portType>
 <binding name = "eBayWatcherBinding" type = "eBayWatcherPortType">
    <soap:binding style = "rpc" transport = http://schemas.xmlsoap.org/soap/http"/>
    <operation name = "getCurrentPrice">
     <soap:operation soapAction=""/>
      <input>
       <soap:body use = "encoded" namespace = "urn:xmethods-EbayWatcher"
                  encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
       <soap:body use = "encoded" namespace = "urn:xmethods-EbayWatcher"
                   encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
 </binding>
 <service name = "eBayWatcherService">
  <documentation>Checks current high bid for an eBay auction</documentation>
  <port name = "eBayWatcherPort" binding = "eBayWatcherBinding">
   <soap:address location ="http://services.xmethods.net:80/soap/servlet/rpcrouter"/>
  </port>
 </service>
</definitions>
```

WSDL Example (priceCheck)

```
<?xml version="1.0"?>
<definitions name="PriceCheck"
   targetNamespace="http://www.skatestown.com/services/PriceCheck"
   xmlns:pc="http://www.skatestown.com/services/PriceCheck"
   xmlns:avail="http://www.skatestown.com/ns/availability"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap"
   xmlns="http://schemas.xmlsoap.org/wsdl/">
   <types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     targetNamespace="http://www.skatestown.com/ns/availability"
     <xsd:complexType name="availabilityType"
      <xsd:sequence>
              <xsd:element name="sku"    type="xsd:string"/>
              <xsd:element name="price"  type"="xsd:double"/>
              <xsd:element name="quantityAvailable"
                                         type="xsd:integer"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:schema>
   </types>
```

WSDL Example (priceCheck)

```
<message name="PriceCheckRequest">
   <part name="sku" type="xsd:string"/>
</message>
<message name="PriceCheckResponse">
   <part name="result" type="avail:availabilityType>
</message>

<portType name="PriceCheckPortType">
   <operation name="checkPrice">
        <input message="pc:PriceCheckRequest"/>
        <output message="pc:PriceCheckResponse"/>
   </operation>
</portType>
```

WSDL Example (priceCheck)

```
<binding name="PriceCheckSOAPBinding"
   type="pc:PriceCheckPortType">
 <soap:binding style="rpc"
               transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="checkPrice">
   <soap:operation soapAction=""/>
   <input>
    <soap:body use="encoded"
      namespace="http://www.skatestown.com/services/PriceCheck"
       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
     <soap:body use="encoded"
       namespace="http://www.skatestown.com/services/PriceCheck"
       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
 </operation>
</binding>
```

WSDL Example (priceCheck)

```
<service name="PriceCheckService">
 <port name="PriceCheck" binding="pc:PriceCheckSOAPBinding">
   <soap:address location=
              http://www.skatestown.com/services/PriceCheck/>
   </port>
</service>
```

StockAvailabilityNotification

```
<?xml version="1.0" ?>
<definitions name="StockAvailableNotification" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   targetNamespace="http://www.skatestown.com/services/StockAvailableNotification"
   xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:reg="http://www.skatestown.com/ns/registrationRequest"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
 <types>
  <xsd:schema targetNamespace="http://www.skatestown.com/ns/registrationRequest"
   xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
   xmlns="http://www.skatestown.com/schemas/ns/registrationRequest">
    <xsd:complexType name="ArrayOfItem">
       <xsd:complexContent>
        <xsd:restriction base="soapenc:Array">
         <xsd:attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]"/>
        </xsd:restriction>
       </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="registrationRequest">
    <xsd:sequence>
     <xsd:element name="items" type="reg:ArrayOfitem”>
     <xsd:element name="address" type="xsd:string"/>
     <xsd:element name="transport" default="http://schemas.xmlsoap.org/soap/smtp" minOccurs="0”>
      <xsd:simpleType>
       <xsd:restriction base="xsd:string">
        <xsd:enumeration value="http://schemas.xmlsoap.org/soap/http"/>
        <xsd:enumeration value="http://schemas.xmlsoap.org/soap/smtp"/>
       </xsd:restriction>
      </xsd:simpleType>
     </xsd:element>
     <xsd:element name="clientArg" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
   </xsd:complexType>
   <xsd:simpleType name="correlationID">
    <xsd:restriction base="xsd:string">…</xsd:restriction>
   </xsd:simpleType>
   <xsd:element name="Expiration” type="xsd:dateTime” />
   <xsd:element name="ErrorString” type="xsd:string” />
  </xsd:schema>
 </types>
```

StockAvailabilityNotification

```
<!-- Message definitions -->
<message name="StockAvailableRegistrationRequest">
   <part name="registration" type="reg:registrationRequest"/>
   <part name="expiration" element="reg:Expiration"/>
</message>
<message name="StockAvailableRegistrationResponse">
   <part name="correlationID" type="reg:correlationID"/>
</message>
<message name="StockAvailableRegistrationError">
   <part name="errorString" element="reg:ErrorString" />
</message>
<message name="StockAvailableExpirationError">
   <part name="errorString" element="reg:ErrorString" />
</message>
<message name="StockAvailableNotification">
   <part name="timeStamp" type="xsd:timeInstant"/>
   <part name="correlationID" type="reg:correlationID"/>
   <part name="items" type="reg:items"/>
   <part name="clientArg" type="xsd:string"/>
</message>
<message name="StockAvailableExpirationNotification">
   <part name="timeStamp" type="xsd:timeInstant"/>
   <part name="correlationID" type="reg:correlationID"/>
   <part name="items" type="reg:ArrayOfItem"/>
   <part name="clientArg" type="xsd:string"/>
</message>
<message name="StockAvailableCancellation">
   <part name="correlationID" type="reg:correlationID"/>
</message>
```

StockAvailabilityNotification

```
<!-- Port type definitions -->
   <portType name="StockAvailableNotificationPortType">
      <!--Registration Operation -->
      <operation name="registration">
         <input message="StockAvailableRegistrationRequest"/>
         <output message="StockAvailableRegistrationResponse"/>
         <fault message="StockAvailableRegistrationError"
            name="StockAvailableNotificationErrorMessage"/>
         <fault message="StockAvailableExpirationError"
            name="StockAvailableExpirationError"/>
      </operation>
      <!--Notification Operation -->
      <operation name="notification">
         <output message="StockAvailableNotification"/>
      </operation>
      <!--Expiration Notification Operation -->
      <operation name="expirationNotification">
         <output message="StockAvailableExpirationNotification"/>
      </operation>
      <!--Cancellation Operation -->
      <operation name="cancellation">
         <input message="StockAvailableCancellation"/>
      </operation>
   </portType>
```

StockAvailabilityNotification

```
<binding name="StockAvailableNotificationSOAPBinding"
         type="StockAvailableNotificationPortType">
 <soap:binding style="rpc“ transport="http://schemas.xmlsoap.org/soap/http"/>
 <documentation>Note: the requestor must invoke registration first.
 </documentation>
 <operation name="registration">
  <soap:operation soapAction=
   "http://www.skatesTown.com/StockAvailableNotification/registration">
   <input>
     <soap:header message="StockAvailableRegistrationRequest"
        part="expiration” use=”literal“>
      <soap:headerfault message="StockAvailableExpirationError"
        part="errorString" use="literal”>
     </soap:header>
     <soap:body part="registration" use="encoded"
        namespace="http://www.skatestown.com/ns/registrationRequest"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/”>
   </input>
   <output>
      <soap:body use="encoded“
       namespace="http://www.skatestown.com/ns/registrationRequest"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   </output>
   <fault name="StockAvailableNotificationErrorMessage">
    <soap:fault name="StockAvailableNotificationErrorMessage"
     namespace="http://www.skatestown.com/ns/registrationRequest"
     encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   </fault>
  </operation>
```

StockAvailabilityNotification

```
  <operation name="notification">
   <output>
    <soap:body use="encoded“
      namespace="http://www.skatestown.com/ns/registrationRequest"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   </output>
  </operation>
  <operation name="cancellation">
   <soap:operation soapAction=
        http://www.skatesTown.com/StockAvailableNotification/cancellation">
    <input>
     <soap:body use="encoded"
       namespace="http://www.skatestown.com/ns/registrationRequest"
       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
  </operation>
 </binding>
 <!-- Service definition -->
 <service name="StockAvailableNotificationService">
  <port name="StockAvailableNotificationPort“
   binding="StockAvailableNotificationSOAPBinding ">
   <soap:address location=
        "http://www.skatestown.com/services/StockNotification"/>
  </port>
 </service>
</definitions>
```

SOAP over SMTP

```
<binding name="PriceCheckSMTPBinding"
   type="pc:PriceCheckPortType">
 <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/smtp"/>
  <operation name="notification">
    <soap:operation soapAction=
       "http://www.skatestown.com/services/PriceCheck"/>
    <input>
     <soap:body use="literal" />
    </input>
    <output>
     <soap:body use="literal" />
   </output>
 </operation>
</binding>
```

WSDL extensions

```
<binding name="priceCheckHTTPGetBinding
                type="pc: priceCheckPortType">
   <http:binding verb="GET"/>
   <operation name="checkPrice">
        <http:operation location="checkPrice"/>
        <input>
                <http:urlEncoded/>
        </input>
        <output>
                <mime:content type="text/xml"/>
        </output>
   </operation>
</binding>
```

http://www.skatestown.com/checkPrice?sku=xxx1234

WSDL with MIME

```
<message name="PriceCheckResponse">
   <part name="result" type="avail:availability"/>
   <part name="picture" type="xsd:binary"/>
</message>

<binding name="PriceCheckSOAPBinding" type="PriceCheckPortType">
…
   <output>
    <mime:multupartRelated>
        <mime:part>
          <soap:body use="encoded"
            namespace="http://www.skatestown.com/services/PriceCheck
            encodingStyle="http://schema.xmlsoap.org/soap/encoding/">
        </mime:part>
        <mime:part>
            <mime:content part="picture" type="image/gif"/>
            <mime:content part="picture" type="image/jpeg"/>
        </mime:part>
    </mime:multupartRelated>
   </output>
…
```

WSDL and Java

- **`portType`** maps into Java interface and name of **`portType`** is typically name of interface

```
public interface PriceCheckPortType extends
java.rmi.Remote {
```

- Package named from `targetNamespace` of definitions element

```
package com.skatestowm.www.services.PriceCheck;
```

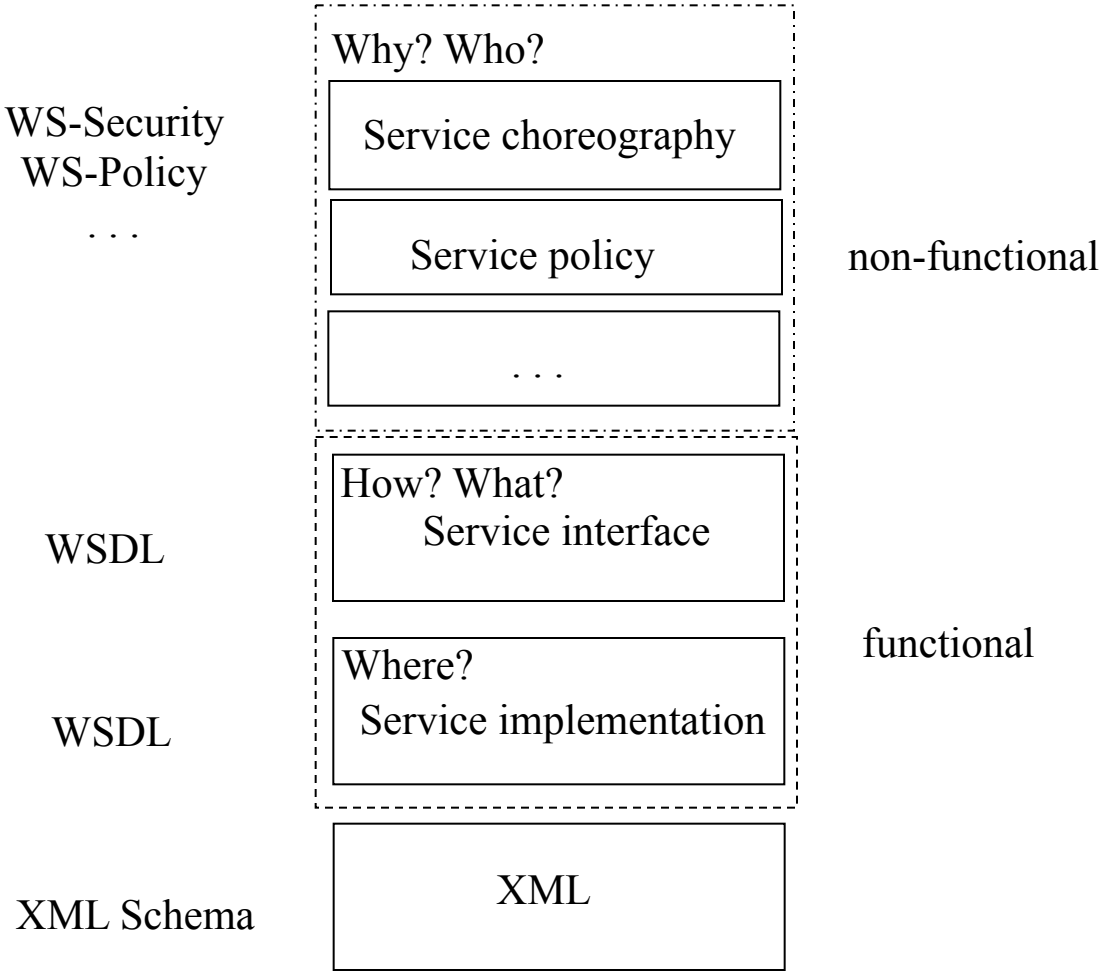
- For each operation a public method is declared as a part of interface
- The signature is built from operation name, input, output and fault elements (faults are included as exceptions)

```
public
com.skatestown.www.ns.availability.AvailableType
  checkPrice(java.lang.String Sku) throws
   java.rmi.RemoteException;
```

WSDL and Java

- For messages referred by parameters elements a separate class is generated for **`complexTypes`**. The name of class is taken from the name of type or element.
- **`binding`** generates a stub class (name of binding – name of class, package – from **`targetNamespace`** of the definition element)
- **`service`** generates interface and class (details about invoking the service from the client application)

What Should Service Describe?



Non-functional descriptions

Policy specification:

- The framework
- The assertions

```
<wsrm:DeliveryAssurance Value="wsrm:ExactOnce"/>

<wsrm:Integrity wsp:Usage="wsp:Required" >
. . .
</wsrm:Integrity>
```

- The attachments

WS Policy

WS policy framework:

- An XML element to act as a container of policy assertions
- A set of XML elements that describe how the policy assertions in the container to be combined
- A set of standard XML attributes that may be associated with policy assertions

WS-Policy (policy container)

```
<wsp:policy ((Name=“ “ TargetNamespace=“ “?)| ID=“ “)
   <policy-specific assertion>*
   <policy-specific security>?
</wsp:Policy>
```

WS Policy (policy operators)

- **`All`** (all assertions are in effect)
- **`ExactlyOne`** (one assertion is in effect)
- **`OneOrMore`**
- The basic **`policy`** element (the same semantics as **`All`**)

WS Policy (policy operators)

```
<wsp:policy Name="PolicyOne"
      TargetNamespace="http://www.skatestown.com/policies">
   <wsp:All>
       <Assertion:A />
       <Assertion:B />
       <Assertion:C />
   </wsp:All>
</wsp:Policy>
```

WS Policy (**`usage`**)

- **`Required`** – must be applied or an error occurs
- **`Rejected`** – must not apply or error occurs
- **`Optional`** – may apply but doesn't have to
- **`Observed`** – the usage is informational
- **`Ignored`** – the usage is informational

WS Policy (**preference**)

```
<wsp:policy Name="PolicyTwo"
   TargetNamespace="http://www.skatestown.com/policies">
   <wsp:ExactOne>
        <Assertion:A wsp:Preference="100" />
        <Assertion:B wsp:Preference="50" />
        <Assertion:C wsp:Preference="1" />
   </wsp:ExactOne>
</wsp:Policy>
```

WS Policy (referencing)

```
<wsp:policy Name="PolicyThree"
   TargetNamespace="http://www.skatestown.com/policies">
  <wsp:ExactOne>
       <wsp:PolicyReference Ref="tns:PolicyTwo />
       <Assertion:D wsp:Preference="10" />
       <Assertion:E wsp:Preference="20" />
  </wsp:ExactlyOne>
</wsp:Policy>
```

WS-Policy document

- Normalization
- Merging
- Intersection

Normal form

```
<wsp:Policy ... >
   <wsp:ExactlyOne>
        [<wsp:All> [ <assertion ...>... </assertion>]*
        </wsp:All> ]*
   </wsp:ExactlyOne>
</wsp:Policy>
```

Example normalization

```
<wsp:Policy ...>
   <wsp:All>
         <wsp:ExactlyOne>
                  <nsSecurityAssertion wsp:Optional="true"/>
                  <nsReliableMessagingAssertion/>
         </wsp:ExactlyOne>
         <nsTransactionAssertion/>
         <nsAuditAssertion/>
   </wsp:All>
</wsp:Policy>
```

```
<wsp:ExactlyOne>
   <wsp:All>
         <nsSecurityAssertion/>
         <nsTransactionAssertion/>
         <nsAuditAssertion/>
   </wsp:All>
   <wsp:All>
         <nsTransactionAssertion/>
         <nsAuditAssertion/>
   </wsp:All>
   <wsp:All>
         <nsReliableMessagingAssertion/>
         <nsTransactionAssertion/>
         <nsAuditAssertion/>
   </wsp:All>
</wsp:ExactlyOne>
```

Merge

Policy P1

```
<wsp:Policy wsu:id="P1"...>
    <wsp:ExactlyOne>
        <wsp:All>
            <nsSecurityAssertion/>
        </wsp:All>
        <wsp:All>
            <nsReliableMessagingAssertion/>
        </wsp:All>
    </wsp:ExactlyOne>
</wsp:Policy>
```

Policy P2

```
<wsp:Policy wsu:id="P2"...>
    <wsp:ExactlyOne>
        <wsp:All>
            <nsTransactionAssertion/>
        </wsp:All>
        <wsp:All>
            <nsAuditAssertion/>
        </wsp:All>
    </wsp:ExactlyOne>
</wsp:Policy>
```

Merge

Policy P1 merged with P2

```
<wsp:Policy wsu:Id="P1_Merged_with_P2"...>
    <wsp:ExactlyOne>
        <wsp:All>
            <nsSecurityAssertion/>
            <nsTransactionAssertion/>
        </wsp:All>
        <wsp:All>
            <nsSecurityAssertion/>
            <nsAuditAssertion/>
        </wsp:All>
        <wsp:All>
            <nsReliableMessagingAssertion/>
            <nsTransactionAssertion/>
        </wsp:All>
        <wsp:All>
            <nsReliableMessagingAssertion/>
            <nsAuditAssertion/>
        </wsp:All>
    </wsp:ExactlyOne>
</wsp:Policy>
```

Intersection

Service provider policy

```
<wsp:Policy wsu:Id="Provider_Policy"...>
   <wsp:ExactlyOne>
        <wsp:All>
               <nsSecurityAssertion level="high"/>
               <nsReliableMessagingAssertion/>
        </wsp:All>
        <wsp:All>
               <nsSecurityAssertion level="medium"/>
               <nsTransactionAssertion/>
        </wsp:All>
   </wsp:ExactlyOne>
</wsp:Policy>
```

Service requestor policy

```
<wsp:Policy wsu:Id="Requester_Policy"...>
   <wsp:ExactlyOne>
        <wsp:All>
               <nsSecurityAssertion/>
               <nsReliableMessagingAssertion timeout="100"/>
               <nsReliableMessagingAssertion retries="3"/>
        </wsp:All>
   </wsp:ExactlyOne>
</wsp:Policy>
```

Intersection

```
<wsp:All>
   <nsSecurityAssertion level="high/>
   <nsSecurityAssertion/>
   <nsReliableMessagingAssertion timeout="100"/>
   <nsReliableMessagingAssertion retries="3"/>
   <nsReliableMessagingAssertion/>
</wsp:All>
```

Special policy

Empty policy:

```
<wsp:Policy ... >
  <wsp:ExactlyOne>
     <wsp:All/>
  </wsp:ExactlyOne>
</wsp:Policy>
```

null policy:

```
<wsp:Policy ... >
  <wsp:ExactlyOne/>
</wsp:Policy>
```

WS Policy (standard assertions)

- *Text Encoding* – lets you declare which character set to use for WS text
- *Language* – allows to describe which human language to be used
- *Spec* – allows define which versions of specification WS is compliant with
- *Message Predicate* – presents details about contents of a message

WS Policy (standard assertions)

```
<wsp:Policy name="SkatestownLanguages"
   targetNamespace=""http://www.skatestown.com/policies">
   <wsp:OneOrMore>
       <wsp:Language Language="en" />
       <wsp:Language Language="es" />
       <wsp:Language Language="fr" />
   </wsp:OneOrMore>
</wsp:Policy>
```

WS Policy (attachments)

- As a part of a policy subject's definition
  - **`PolicyRefs`** or **`PolicyURL`** attributes
  - Policy inheritance and merging
- Externally to the subject's definition
  - **`PolicyAttachment`** element

WS Policy (attachments)

```
<service name="PriceCheck"
  wsp:PolicyRefs="stp:SkatestownLanguages"
  xmlns:wsp="http//schemas.xmlsoap.org/ws/2002/12/policy"
  xmlns:stp="http://www.skatetowm.com/policies">
  <port name="PriceCheck"
                  binding:"pc:PriceCheckSOAPBinding">
  . . .
```

WSDL 2.0 (interface)

- **`interface`** replaces **`portType`**
- Introduced interface extension
  - Works similar to interface inheritance in Java

```
<interface name="StockAvailableNotification"
  extends="sa:Notification" . . .>
    <operation name="unregistration" . . .>
```

- **`styleDefault`** attribute added - describes a style of message exchange pattern applied to all child **`operations`** in **`interface`**
- Features and properties
  - Similar to feature and properties in SOAP

WSDL 2.0 (operation)

- Clarification of transmission primitives
  - Instead of 4 in WSDL 1.1 it is possible to define message exchange pattern
  - **`style`** attribute takes URI name of message exchange pattern
  - Most practical are **`in-out`** and **`in-only`** patterns

```
<interface name="PriceCheck">
  <operation name="checkPrice"
   style="http://www.w3.org/2003/11/wsdl/in-out">
  …
  </operation>
</interface>
```

WSDL 2.0 (operation)

- Elimination of **`message`** and **`part`** elements
  - Names of XML elements are used directly

```
<input message="avail:sku"/>
```
instead of
```
<input message=" pc:PriceCheckRequest"/>
```
and
```
<message name="PriceCheckRequest">
    <part name="result" element="avail:sku"/>
```

- Operation overloading is forbidden
- **`fault`** element is divided into **`infault`** and **`outfault`**

WSDL 2.0 (types)

- Clarifies contents of types as either XML **`schema`** element or XML **`import`** element
- Still allows usage of other type systems besides XML schema

WSDL 2.0 (binding)

- Structural changes
- Used also for binding defined features and properties
- Replaces SOAP 1.1 binding type with SOAP 1.2 type

```
xmlns:soap="http://www.w3.org/2003/11/wsdl/soap12"
…
<binding…
   <soap:binding protocol="http://www.w3.org/2003/051/soap/
   bindings/HTTP/"  />
</binding>
```

WSDL 2.0 (endpoint and service)

- **`endpoint`** is just renaming of the **`port`**
- Added **`interface`** attribute specifying which **`interface`** is implemented by the service
- All **`endpoints`** child's elements should refer to the same **`interface`**

WSDL 2.0 (definitions, import and include)

- The **`name`** attribute is removed from definitions
- The include element is introduced
  - The **`import`** element is used to import WSDL definitions from other WSDL namespaces
  - The **`include`** element is used to include elements defined in different documents but in the same namespace

WSDL 2.0 PriceCheck

```
<?xml version="1.0"?>
<definitions
   targetNamespace="http://www.skatestown.com/services/PriceCheck"
   xmlns:pc="http://www.skatestown.com/services/PriceCheck"
   xmlns:avail="http://www.skatestown.com/ns/availability"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
   www.w3.org/2003/11/wsdl/soap12"
   xmlns:wsdl12="http://www.w3.org/2003/11/wsdl"
   xmlns="http://www.w3.org/2003/11/wsdl">
   <types>
        <xsd:schema
            targetNamespace="http://www.skatestown.com/ns/availability"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            <xsd:element name="sku" type="xsd:string" />
            <xsd:complexType name="availabilityType"
                <xsd:sequence>
                  <xsd:element ref="avail:sku"/>
                  <xsd:element name="price" type"="xsd:double"/>
             <xsd:element name="quantityAvailable"
                                type="xsd:integer"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:element name="stockAvailability"
                          type="avail:availabilityType" />
        </xsd:schema>
   </types>
```

WSDL 2.0 PriceCheck

```
<interface name="PriceCheck">
   <operation name="checkPrice"
            style=http://www.w3.org/2003/11/wsdl/in-out">
     <input  message="avail:sku"/>
     <output message="avail:StockAvailability"/>
   </operation>
</interface>
<binding name="PriceCheckSOAPBinding" type="pc:PriceCheck">
   <soap:binding  protocol="http://www.w3.org/2003/05/soap/
     bindings/HTTP"/>
</binding>
<service name="PriceCheck" interface="pc:PriceCheck">
   <endPoint name="PriceCheck"
             binding="pc:PriceCheckSOAPBinding">
     <soap:address location=
           http://localhost:8080/axis/services/PriceCheck/>
   </endPoint>
</service>
</definitions>
```

Summary

- Main features of WSDL
- WS-Policy
- WSDL and Java
- WSDL 2.0

Next lecture

- UDDI

  Text-book **Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, 2nd Edition**

Chapter 6