WSDL 2.0

Marlon Pierce
Community Grids Lab
Indiana University
WSDL 1.1 to WSDL 2.0

• WSDL 1.1 never actually became a full fledged recommendation.
• WSDL 2.0 working draft has just completed public comment phase.
  – September 19th
• In general, WSDL 2.0 seems to clean up a lot of WSDL 1.1’s complications.
• But really for most users this will not be an issue, since tools such as Apache Axis 2 will conceal these issues.
A WSDL Example

• Acknowledgement: I took this from the WSDL 2.0 Primer

• Basic parts:
  – Description: root tag
  – Types: local data types
  – Interface: the new portType
  – Binding: bind the interface to transport
  – Service: bind the binding to an end point.

• But first, a big picture.
No more `<message/>`

`<portType>` is now `<interface>`

WSDL 2.0 InfoSet Diagram

http://www.w3.org/TR/2005/WD-wsdl20-primer-20050803/
The WSDL Sandwich

<?xml version="1.0" encoding="utf-8" ?>

<description xmlns="http://www.w3.org/2005/08/wsd1"
     targetNamespace= "http://greath.example.com/2004/wsd1/resSvc"
     xmlns:tns="http://greath.example.com/2004/wsd1/resSvc"
     xmlns:wsoap= "http://www.w3.org/2005/08/wsd1/soap"
     xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
     xmlns:wsdlx= "http://www.w3.org/2005/08/wsd1-extensions">

    <documentation>Blah blah blah </documentation>
    <!-- Types, Interface, Binding, and Service -->

</description>
<Types> Example

<types>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="http://greath.example.com/2004/schemas/resSvc">
    <xs:element name="checkAvailability" type="tCheckAvailability"/>
    <xs:complexType name="tCheckAvailability">
      <xs:sequence>
        <xs:element name="checkInDate" type="xs:date"/>
        <xs:element name="checkOutDate" type="xs:date"/>
        <xs:element name="roomType" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:element name="checkAvailabilityResponse" type="xs:double"/>
    <xs:element name="invalidDataError" type="xs:string"/>
  </xs:schema>
</types>
Notes on `<types>`

- `<type/>` technically includes a sequence of `<xsd:any>` tags
  - The schema defines it this way.
- In practice, you just use XML Schemas to define messages.
- The `<input>` and `<output>` tags of the interface operations use these element.
Example

```xml
<interface name = "reservationInterface" >
  <fault name = "invalidDataFault" element = "ghns:invalidDataError"/>
  <operation name="opCheckAvailability"
    pattern="http://www.w3.org/2005/08/wSDL/in-out"
    style="http://www.w3.org/2005/08/wSDL/style/iri"
    wsdlx:safe = "true">
    <input messageLabel="In" element="ghns:checkAvailability" />
    <output messageLabel="Out"
      element="ghns:checkAvailabilityResponse" />
    <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
  </operation>
</interface>
```
Notes on `<interface/>`

- Several new features over portType.
- Interfaces now contain `<fault>` definitions as well as `<operation>`.
- `<operation>` has three important attributes
  - pattern (required): the URI of the appropriate message pattern
    - Out-in, in-out, in-only, out-only
    - But not up-down or round-round
  - Style (optional): the URI name of the encoding convention to be used.
    - RPC, IRI, Multipart
  - `wsdl:safe` (optional): if true, means the client acquires no additional obligations.
    - Safe pages can be pre-fetched and cached for performance.
- The `<input>` and `<output>` tags all refer to the `<type>` definition.
<binding/> and <service/>

```xml
<binding name="reservationSOAPBinding"
    interface="tns:reservationInterface"
    type="http://www.w3.org/2005/08/wsd1/soap"
    wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">
    <fault ref="tns:invalidDataFault" wsoap:code="soap:Sender"/>
    <operation ref="tns:opCheckAvailability"
        wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
</binding>

<service name="reservationService"
    interface="tns:reservationInterface">
    <endpoint name="reservationEndpoint"
        binding="tns:reservationSOAPBinding"
        address ="http://greath.example.com/2004/reservation"/>
</service>
```
Notes on `<binding/>`

- The binding element’s “interface” attribute connects it to the appropriate `<interface/>`.
  - Recall there can be more than one `<interface/>`.
  - This is a non-reusable binding example.
  - Bindings that omit “interface” attribute are *reusable*.
    - Must also omit operation and fault specific details.
  - What’s left?
    - See `wsoap:code` and `wsoap:mep` from previous example.

- The child `<operation/>` tag tells you which transport protocol to use with the associated interface operation.
Notes on `<service/>`

- The `<service/>` is associated with an `<interface/>` by QName through the “interface” attribute.
- The `<endpoint/>` tag associates with a `<binding/>` through the “binding” attribute.
- The “address” attribute is the URL of the actual service.
- So a service can have multiple endpoints, each binding to a different transport protocol, but all associated with the same interface.
  - So your interface can support SOAP 1.1 and SOAP 1.2 bindings through two different endpoints.
interface Inheritance

• WSDL 2.0 <interface/> elements can use the “extend” attribute to inherit operations.
• Extended interfaces gain all operations from the other interface.
• Two caveats
  – Recursive loops are forbidden.
  – All operations should have unique QNames.
  – If two operations have the same QName, they MUST be identical operations, or there is an error.
• Recall that <interface/> can occur zero or more times, so we can use “extends” and multiple interfaces to “universal” operations.
  – In following example, the reservationInterface will also have the opLogMessage operation.
An Interface Extension Example

```xml
<inteface name = "messageLogInterface">
  <operation name="opLogMessage"
    pattern="http://www.w3.org/2005/08/wsd1/out-only">
    <output messageLabel="out" element="ghns:messageLog" />
  </operation>
</interface>

<inteface name="reservationInterface" extends="tns:messageLogInterface">
  <operation name="opCheckAvailability"
    pattern="http://www.w3.org/2005/08/wsd1/in-out"
    style="http://www.w3.org/2005/08/wsd1/style/iri"
    wsdlx:safe = "true">
    <input messageLabel="In" element="ghns:checkAvailability" />
    <output messageLabel="Out"
      element="ghns:checkAvailabilityResponse" />
    <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
  </operation>
</interface>
```
More Stuff: Features

• WSDL 2.0 interfaces have optional `<feature/>` tags.
• These seem to be related to “quality of service” considerations.
  – That is, should the operation be secure, reliable, etc?
  – These correspond roughly to various Web Service extensions.
    • And corresponding SOAP headers.
  – But the actual connection seems to be very loose.