ISSGC’05

Web Services Descriptions and SOAP messages

Richard Hopkins,
National e-Science Centre, Edinburgh
Goals –
- To be able to understand
  - WSDL definition for a standard SOAP binding
  - A soap message

- **Structure**
  - SOAP Messages
  - General Structure of WSDL
  - Details of abstract Service Definition
  - Details of Physical Service Definition - Core
  - Physical Service Definition - Extensions
“SOAP” what it is

• **Name**
  – Originally – Simple Object Access Protocol
  – Temporarily – Service Oriented Architecture Protocol ?
  – Now (SOAP 1.2) – Not an acronym

• **Purpose**
  – A extensible protocol to enable the exchange of
    ▪ structured and typed information
    ▪ between peers
    ▪ in a decentralised, distributed environment

• **Status**
  – **SOAP 1.2** – [http://www.w3.org/TR/soap12-part0](http://www.w3.org/TR/soap12-part0)
    ▪ W3C recommendation, June 2003
  – **SOAP 1.1** – [http://www.w3.org/TR/NOTE-SOAP-20000508](http://www.w3.org/TR/NOTE-SOAP-20000508)
    ▪ W3C submission May 2000 – but that’s what people use currently
Main Architectural Features

- **XML based**
- **Higher order Protocol** –
  - Built on some underlying protocol - binding
    - Extensibility – can define binding for any underlying protocol
    - Usually HTTP – a specific standard extension
- **Single Message Protocol**
  - Defines standard for a single message communication
  - Multi-message conversations require a means to associate one message with another
    - Via underlying protocol (e.g. use of same connection)
    - Via the application (specific message-id information as part of the soap message)
- **Multi-stage message processing** –
  - The soap Processing model
Each SOAP message will have:

- **Outer layer(s) for underlying protocols**
  - Only consider HTTP
- **Envelope (XML root element)**
- **Header (optional)**
  - Multiple header blocks/entries
  - For different purposes – factorisation
  - For different processing stages
    - Actors
- **Body (mandatory)**
  - The payload
  - Zero or more XML elements
  - May be a Fault element
    - Specific fault reporting standard
XML Message Representation

<table>
<thead>
<tr>
<th>Outer protocol – HTTP headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP message is an XML document</td>
</tr>
<tr>
<td>XML root element is SOAP envelope</td>
</tr>
<tr>
<td>“<a href="http://schemas.xmlsoap.org/soap/envelope/%E2%80%9D">http://schemas.xmlsoap.org/soap/envelope/”</a> Identifies SOAP namespace and the SOAP version being used</td>
</tr>
<tr>
<td>Identify application namespaces</td>
</tr>
<tr>
<td>Two headers</td>
</tr>
<tr>
<td>The body</td>
</tr>
</tbody>
</table>

```xml
<?xml version="1.0"?>
<env:Envelope
  xmlns:env="...xmlsoap.org..."
  xmlns:s="http://ex.org/soap"
  xmlns:c="http://ex.org/customer">
  <env:Header>
    <s:authentication
      env:actor="...ex.org...">
      <c:username>Fred</c:username>
      <c:password>yhjik154</c:password>
    </s:authentication>
    <s:authorisation
      env:actor="...ex.org...">
      <c:accountNumber>17-365-37a</c:accountNumber>
    </s:authorisation>
  </env:Header>
  <env:Body>
    <c:PurchaseOrder>....</c:PurchaseOrder>
  </env:Body>
</env:Envelope>
```
Types

- Simple value in a SOAP message will have a type as in Schemas standard, which defines their lexical form.
- Soap provides two compound types:
  - Se:Struct – components are uniquely named
  - Se:Array – components are identified by position
- Array is of type SEnc:Array or some derivative thereof.
- Can specify shape and component type using attribute SEnc:arrayType.

```
<element name="A" type="SEnc:Array"/>
```

```
<A SEnc:arrayType="xsd:integer [2,3] [2]">
  <A1>
    <n>111</n> <n>112</n> <n>113</n>
    <n>121</n> <n>122</n> <n>123</n>
  </A1>
  <A2>
    <n>211</n> <n>112</n> <n>213</n>
    <n>221</n> <n>122</n> <n>223</n>
  </A2>
</A>
```

- [2] - An array of 2 elements -
- [2,3] Each is a 2 x 3 array of
- Xsd:integer – standard schema type.
Faults reported in the body – single element
Zero or more header entries –
for detail error information pertaining to original header entries

Body
Fault
faultcode
faultstring
faultactor
detail ?

a Qname, e.g env:mustUnderstand
Human readable text
Actor that was operating (URI)
Optional
(default = ultimate destination)

Any structure of further application-specific information
Its presence means body was processed

SOAP ENVELOPE
SOAP Header
HEADER ENTRY
HEADER ENTRY
...
SOAP BODY
Fault

Transport protocol
HTTP header
Fault Message Example

```
<env:Envelope ...
<env:Body>
<env:Fault>
  <env:faultcode>env:Server</env:faultcode>
  <env:faultstring>Server Error</env:faultstring>
<env:detail
  xmlns:f="http://ex.org/faults"
  env:encodingStyle="...">  
    <f:faultdetail1>
      <f:faultcode>129</f:faultcode>
      <f:excuse> not my fault really </f:excuse>
    </f:faultdetail1>
    <f:faultdetail2> .... </f:faultdetail2>
```

- Standard error code
- Explanation
- Application-specific Error code
- Explanation
The Standard Fault Codes

- **env:VersionMismatch**
  - Un-recognised namespace for the env:Envelope
- **env:MustUnderstand**
  - A mandatory header entry was not understood
- **env:Client**
  - It’s your fault (e.g. wrong info. In body); re-send won’t work.
  - Must have detail element
- **env:Server**
  - It’s our fault (e.g an upstream processing node not responding).
  - Might succeed if sent later.
  - Can have detail element
Goals –
  – To be able to understand
    ▪ WSDL definition for a standard SOAP binding
    ▪ A soap message

• Structure
  – SOAP Messages
  – General Structure of WSDL
  – Details of abstract Service Definition
  – Details of Physical Service Definition - Core
  – Physical Service Definition - Extensions
What is WSDL?

- An XML format
- For describing network services
  - Operates either on
    - Documents
    - Procedure calls
  - Describes the exposed interface – what the consumer sees of the service
  - Constitutes a contract with the client
  - Provides a specification of what is offered by the service provider which can be relied on by the service consumer
- Supports Separation of concerns
  - abstract structure – operations and messages
  - binding to a specific underlying protocol
  - definition of a particular deployed service
  - To allow common definition and re-combination
- Here using WSDL 1.1 – a W3C submission (March 2001)
  - 2.0 – is a last call working draft (Aug 2004) – many differences
• Example

• Company Provides two types of Service (PortTypes)
  – General Service
    ▪ Get general information (GenInfo)
    ▪ Open an Account (OpenAcc)
  – Customers Service (being a “Customer” = having an account)
    ▪ Purchase Order (PurchOrder)
    ▪ Invoice (Inv)
    ▪ Payment Advice (PayAdv)
    ▪ Get Statement (GetStmt)
    ▪ Notify overdue payment (Overdue)

• Both over two kinds of binding
  – Web - HTTP
  – Email - SMTP
Abstract Definition - PortTypes

- Start with PortType = Interface
- Set of operations
- For each operation, a number of messages – input; output; faults
- Message is defined separately

  as a number of message parts
  each part has a type (using schema)

PortType: General
Op: GenInfo
Op: OpenAcc

Types: ....

Messages: POin…Error1…

PortType: Customer
Op: PurchOrder
  POin, Error1, ....
Op: GetStmt ...
Op: Inv ....
Op: PayAdv ....
Op: Overdue ....
Physical Definition - Binding

- **Binding** –
  - A binding of a portType to a communication protocol for using it
  - Specifies
    - The portType
    - The underlying protocol(s)
    - How the logical structure is represented using the underlying protocol
  - Here two bindings for each PortType – web, e-mail

```
PortType: General
  Op: GenInfo ↔
  Op: OpenAcc ↔

Types ....

Messages: POin...Error1...

Binding: ..HTTP..

PortType: Customer
  Op: PurchOrder

Binding: ..SMTP..

PortType: General
  Op: GenInfo ↔
  Op: OpenAcc ↔

Binding: ..HTTP..

PortType: Customer
  Op: PurchOrder

Binding: ..SMTP..
```
Physical Definition - Service

- **Service** – defines one or more ports, each with
  - **Location** – URL – here sharing of locations
  - **Binding** and thus **portType**
    - The interface provided by the port
    - how it is realised over a particular protocol
  - Here one service for each portType – there are alternatives ….

**PortType:** General
- **Op:** GenInfo <->
- **Op:** OpenAcc <->

**Message:** POin…Error1…

**Types:** ….

**Binding:** ..HTTP..

**Port:** www. .../WS-web

**Service:** GeneralService

**PortType:** Customer
- **Op:** PurchOrder

**Binding:** ..SMTP..

**Port:** WS-em@ …

**Service:** CustomerService
Partitioning of WSDL

- Could put all definitions in one WSDL file – that’s what is produced by JAX
- For hand-crafted WSDL, could spread over multiple files, e.g. -
  - one WSDL file per service –
    - Gives control over publication use – CustomerService not in public registry
    - Different services may have semi-independent development
  - common message and type definitions – may be shared between interfaces
### Structure

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;import&gt;</code></td>
<td>Incorporate external definitions</td>
</tr>
<tr>
<td><code>&lt;types&gt;</code></td>
<td>Logic structure of data being transmitted</td>
</tr>
<tr>
<td><code>&lt;message&gt;</code></td>
<td>Transmittable messages</td>
</tr>
<tr>
<td><code>&lt;portType&gt;</code></td>
<td>Interface – operations and assoc. messages</td>
</tr>
<tr>
<td><code>&lt;binding&gt;</code></td>
<td>How messages will be transmitted</td>
</tr>
<tr>
<td><code>&lt;service&gt;</code></td>
<td>How a service is accessed</td>
</tr>
<tr>
<td><code>&lt;port&gt;</code></td>
<td>Web-address</td>
</tr>
</tbody>
</table>

**Abstract**

- `<schema>`
  - `<TYPE>`

**Physical**

- `<part>`
- `<operation>`
  - `<MESSAGETYPE>`

**Abstract Physical**

- `<types>`
  - `<types>`
    - `<schema>`
      - `<TYPE>`

---

*ISSGC’05 – June 2005*
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="CustServWSDL">
    <wsdl:targetNamespace>HTTP://ex.org/wsdl/Cust</wsdl:targetNamespace>
    <xmlns:w="HTTP://ex.org/wsdl/Cust">
    <xmlns:wsdl="…xmlsoap.org/wsdl/">
    <xmlns:soap="…xmlsoap.org/wsdl/soap/” …>
    <xmlns:y="Y” …
    <xmlns:t="T” … …” …>
    <import namespace="Y" location="Yloc">
        …
    </import>
    <types>
        <wsdl:schema targetNameSpace="T” …
        … </>
        … </>
        <wsdl:message> … </>
        … </>
        <wsdl:portType> … </>
        … </>
        <wsdl:binding> … </>
        … </>
        <wsdl:service> … </>
        … </>
    </types>
</wsdl:definitions>
Goals –

- To be able to understand
  - WSDL definition for a standard SOAP binding
  - A soap message

• Structure
  - SOAP Messages
  - General Structure of WSDL
  - Details of abstract Service Definition
  - Details of Physical Service Definition - Core
  - Physical Service Definition - Extensions
Types and Messages

TYPES –
- Each type is for one or more message parts
- Need a target namespace, as for any Schema
- A prefix for use in message parts

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl: definitions ....
  xmlns:m = "../mytypes/serviceA" >
  <wsdl: types>
    <xsd: schema ....
      targetNamespace= "../mytypes/serviceA" >
    <xs:complexType name="PreludeT">
      <xs:sequence>
        <xs:element name="Account" type="xs:string"/>
        <xs:element name="Date" type="xs:date"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="POentriesT">
      .... </xs:complexType>
  </xsd: schema>
  </wsdl: types>
</wsdl: definitions>
```

Types
Schema
- PreludeT
  - Account
  - Date
POentriesT ....

Message: POin
  - Part: Prelude
  - Part: POentries

Message: GetStmtIn
  - Part: Prelude

Message: Error1
  - ....
  - ....
Types and Messages

Message

- Has a name – so message can be referenced by a portType definition
- Consists of one or more parts,
  - Each part is a logical unit, e.g. a parameter
    - No parts in WSDL 2.0
  - Has a name so that it can be referenced by a Binding definition
    - E.g. to put one part in a header and the other part in the body
  - Has a type –
    - a Schema type definition or a Schema element definition
    - a standard type – from an imported Schema

```xml
<wsdl:message name="POin">
  <part name="Prelude" type="t:PreludeT">
  </part>
  <part name="POentries" type="t:accInfoT">
  </part>
</wsdl:message>

<wsdl:message name="OverdueIn">
  <part name="excuse" type="xs:string"></part>
</wsdl:message>
```

Input message for the purchase order operation

Defined type – the namespace of the in-line schema

Input (response) message for the Overdue Payment operation

standard type
PortTypes

PORTTYPE – an interface comprising a set of operations
• Organisation of functionality into portTypes and operations is similar to O-O design
• A portType is a coherent unit of exposed functionality – operations make sense together
  – E.g. Currency conversion might be a service used in processing customer transaction
  – But would not expect a convertCurrency operation for this service
• But … larger granularity than O-O
• … Deployment considerations
  – Split between General and Customer may be that say General has a wider range of available bindings/locations
• Each operation declares a number of messages which can be communicated as the interface to the operation
• These messages conform to one of four message exchange patterns – input/output sequencing ….
Four patterns

\[ \rightarrow \text{IN then OUT} \quad \text{Request/Response – most usual} \]
\[ \rightarrow \text{OUT} \quad \text{Notify *} \]
\[ \leftarrow \text{IN} \quad \text{One-way} \]
\[ \leftrightarrow \text{OUT then IN} \quad \text{Solicit/Response *} \]

* Reversed roles
Service Provider
proactive = client
Service Consumer
reactive = server

**PortType:** General
**Op:** GenInfo \( \leftrightarrow \)
**Op:** OpenAcc \( \leftrightarrow \)

**PortType:** Customer
**Op:** PurchOrder \( \leftrightarrow \)
**Op:** GetStmt \( \leftrightarrow \)
**Op:** Inv \( \rightarrow \)
**Op:** PayAdv \( \leftarrow \)
**Op:** Overdue \( \rightarrow \leftrightarrow \)

**Messages:**
POin, POout, Err1, …
InvoiceOut,
PayAdvIn,
ODueOut, ODueResp, Err2,..

**Types:** ….
Message Exchange Patterns

- **Message exchange pattern** is determined by sequence of message declarations
  - $\leftrightarrow$ **Request-Response** –
    - **input**
    - **output**
    - **fault***
  - $\leftrightarrow$ **Solicit-Response** –
    - **output**
    - **input**
    - **fault***
  - **Notify** –
    - **output**
  - $\leftarrow$ **One-way (Request)** –
    - **input**

- **Single message patterns** can’t have fault message
- (in WSDL 2.0 this is explicit and more general – named patterns)

---

**PortType:** Customer
- **Op:** PurchOrder
  - **In:** POin
  - **Out:** POOut
  - **Fault:** Error1 …
- **Op:** Overdue
  - **Out:** OverdueOut
  - **In:** OverdueIn
  - **Fault:** ErrorThreat
- **Op:** Inv
  - **Out:** InvOut
- **Op:** PayAdv
  - **In:** PayAdvIn
Request - Response

• Message to service provider; reply to service consumer; possible fault messages
• A logical pattern, Binding might be e.g. An HTTP request/response or two HTTP requests

```
<wSDL:portType name="CustomerP">
  <wsdl:operation name="PurchOrder">
    <wsdl:input name="PurchOrderRequest" message="w:POin">
    <wsdl:output name="PurchOrderResponse" message="w:POout">
    <wsdl:fault name="Error1" message="w:Error1">
      ...
    </wsdl:operation ...
  <wsdl:portType name="CustomerP"/>
```

Provide a Port/OP. names, to be referenced by Binding
Provide a name for that message in this context, to be referenced by Binding
Default message name – operation + request/response
Refer to a message definition using the WSDL’s target namespace
w xmlns="…/wsdl/…”
Whereas messages use schema namespace
t xmlns="…/types/…”
Solicit - Response

- Message from service provider to consumer; reply from consumer to provider; possible fault messages

```xml
<wsdl:portType name="CustomerP">
    ...
    <wsdl:operation name="Overdue">
        <wsdl:output name="OverdueSolicit" message="w:OverdueOut">
        </wsdl:output>
        <wsdl:input name="OverdueResponse" message="w:OverdueIn">
        </wsdl:input>
        <wsdl:fault name="Threat" message="w:ErrorThreat">
            ...
        </wsdl:fault>
    </wsdl:operation>
</wsdl:portType>
```

Default message name – operation + solicit/response
Single Messages

- Notify
  - Message from service provider to consumer, with no reply
  - Example – “Invoice”
    - Send an invoice
- One-way - Request with no reply
  - Message from service consumer to provider
  - Example – “payment advice”
    - Company gets notification from customer that a payment has been made

```xml
<wSDL:portType name="CustomerP">
  ...
  <wSDL:operation name="Inv">
    <wSDL:output name="Inv" message="w:InvOut" />
    <wSDL:operation name="w:PayAdv">
      <wSDL:input name="PayAdv" message="w:PayAdvIn" />
    </wSDL:operation>
  </wSDL:operation>
</wSDL:portType>
```

Default message name – operation name
Goals –
  – To be able to understand
    ▪ WSDL definition for a standard SOAP binding
    ▪ A soap message

• Structure
  – SOAP Messages
  – General Structure of WSDL
  – Details of abstract Service Definition
  – Details of Physical Service Definition - Core
  – Physical Service Definition - Extensions
A Binding defines:

- A particular PortType – named as its “type”
- Particular message format and communication protocol details
  - By extensibility point
  - A standard extension is SOAP binding
- A binding name, for use in service definition
Structure of Binding Element

<table>
<thead>
<tr>
<th>Binding follows structure of PortType</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;wsdl:portType</code> name=&quot;CustomerP&quot;&gt;</td>
</tr>
<tr>
<td><code>&lt;wsdl:operation</code> name=&quot;PurchOrder&quot;&gt;</td>
</tr>
<tr>
<td><code>&lt;wsdl:input</code> name=&quot;PurchOrderRequest&quot; message=&quot;w:POin&quot;&gt;</td>
</tr>
<tr>
<td><code>&lt;wsdl:output</code> name=&quot;PurchOrderResponse&quot; message=&quot;w:POout&quot;&gt;</td>
</tr>
<tr>
<td><code>&lt;wsdl:fault</code> name=&quot;Error1&quot; message=&quot;w:Error1&quot;&gt;</td>
</tr>
<tr>
<td>..... Other faults</td>
</tr>
<tr>
<td><code>&lt;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;wsdl:operation ...&gt;</code> <code>&lt;/&gt;</code> <code>...</code> <code>&lt;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;wsdl:binding</code> name=&quot;CustWebB&quot; type=&quot;w:CustomerP&quot;&gt;</td>
</tr>
<tr>
<td>... some binding info ...</td>
</tr>
<tr>
<td><code>&lt;wsdl:operation</code> name=&quot;PurchOrder&quot;&gt;</td>
</tr>
<tr>
<td><code>&lt;wsdl:input</code></td>
</tr>
<tr>
<td>... some binding info ... <code>&lt;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;wsdl:output</code></td>
</tr>
<tr>
<td>... some binding info ... <code>&lt;/&gt;</code></td>
</tr>
<tr>
<td><code>&lt;wsdl:fault</code> name=&quot;Error1&quot;&gt;</td>
</tr>
<tr>
<td>... some binding info ... <code>&lt;/&gt;</code></td>
</tr>
<tr>
<td>Multiple faults so need Name - No prefix</td>
</tr>
<tr>
<td><code>&lt;wsdl:operation name=...&gt;</code> <code>...</code> <code>&lt;/&gt;</code></td>
</tr>
<tr>
<td><code>[Image 42x70 to 81x228]</code></td>
</tr>
</tbody>
</table>
Protocol Specific Information

- Binding definition is framework for extension points – a bit intangible

To be referenced by port defn

Ref’s PortType

<wsdl:binding name="CustWebB" type="w:CustomerP">
  ... some binding info ...
</wsdl:binding>

<wsdl:operation name="PurchOrder">
  ... some binding info ...
</wsdl:operation>

<wsdl:input>
  ... some binding info ...
</wsdl:input>

<wsdl:output>
  ... some binding info ...
</wsdl:output>

<wsdl:fault name="Error1">
  ... some binding info ...
</wsdl:fault>

... other faults ...

<wsdl:operation name="Inv">
  ....
</wsdl:operation>

.... Other operations ....

Protocol-specific information applying to all operations

Protocol-specific information applying to both input and output message

1. Protocol specific information applying to particular message
2. How message parts map into protocol and data formats
• Can have multiple services in one WSDL definition document
• Each Service can have multiple ports, each bound to a binding
• For WSDL 2.0 – all ports of a service must have the same portType
  – Can have different portTypes in WSDL 1.1 –
    consumer may need all functionalities for the service to be useful
• Two ports having the same portType means same semantics
• Gives the location, a URL –
  – this is a SOAP extension of WSDL, not WSDL core

```xml
<wsdl:service name="CustomerService">
  <wsdl:port name="CustWebSP" binding="w:CustWebB">
    <soap:address location="http://www.ex.org/WS"/>
  </wsdl:port>
  <wsdl:port name="CustEmailSP" binding="w:CustEmailB">
    <soap:address location="mailto:WS-EM@www.ex.org"/>
  </wsdl:port>
</wsdl:service>
```
Goals –
  – To be able to understand
    ▪ WSDL definition for a standard SOAP binding
    ▪ A soap message

• Structure
  – SOAP Messages
  – General Structure of WSDL
  – Details of Abstract Service Definition
  – Details of Physical Service Definition - Core
  – Physical Service Definition - Extensions
There are a number of defined bindings to be used in the extension points:

- **SOAP** – identifying the SOAP 1.1 standards
  - Transport
    - *Over HTTP*
    - *Over SMTP*
    - *...
  - Style
    - *RPC*
    - *Document*
  - Use
    - *Literal*
    - *Encoded*
- **SOAP over HTTP** is most commonly used
  - all we will deal with here
The Soap Binding Extension

```xml
<wSDL:binding name="CustWebB" type="w:CustomerP">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http">
    <soap:operation name="PurchOrder"> ...
    <soap:operation name="...
    <soap:operation name="...
  </soap:binding>
</wSDL:binding>
```

**PORT LEVEL**
Replaces an ANY extension point in general definition.

**soap:binding** element means Using SOAP standards.
Message structure is

```xml
<soap:envelope>
  <soap:header>...
  <soap:body>...
```

**style=** - Default for all operations;
  = "rpc" – body is parameters/return
  = "document" – body is one document
Optional
default = "document"

**Transport=**
URI to identify some protocol
Optional
The Soap Binding Extension

PORT LEVEL
Replaces an ANY extension point in general definition

OPERATION LEVEL

<wsdl:binding name="CustWebB" type="w:CustomerP">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http">
    <wsdl:operation name="PurchOrder">
      <soap:operation
        soapAction="http://ex.org/PO"
        style="rpc">
        <wsdl:input> ... </>
        <wsdl:operation name="...">...</>
    </soap:operation>
  </soap:binding>
</wsdl:binding>

soapAction =
URI, the value for the HTTP header “SOAPAction”
Mandatory for SOAP/HTTP
For “document” style gives the operation
For JAX-RPC – empty, “”

style =
Over-rides port-level style
The Soap Binding Extension

PORT LEVEL

OPERATION LEVEL

MESSAGE LEVEL

Namespace=
the namespace to be used for validating the outermost elements

Use = “Encoded” : each message part references a type – an abstract type encoded using the scheme(s) identified by encodingStyle

Use = “literal” : each message part ref’s a type or element which gives the concrete format

encodingStyle = “..” a URI list to identify encoding scheme(s)
Soap-encoding is http://schemas.xmlsoap.org/soap/encoding

<wsdl:binding name="CustWebB" type="w:CustomerP">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http" >
  <wsdl:operation name="PurchOrder">
    <soap:operation
      soapAction="http://ex.org/PO"
      style="rpc">
    <wsdl:input>
      <soap:body use="encoded"
        encodingStyle="http://.../encoding/"
        namespace="http://ex.org/wsdl/Cust" /></>
      .... </>
    <wsdl:operation name="...">...</>
    .... </>
  </soap:operation>
</soap:binding>

ISSGC’05 – June 2005
WSDL and SOAP 38
• **RPC**
  - Hint that this is best dealt with as a procedure call (/return)
  - Message parts are parameters which are wrapped as one component of Body
  - As in the SOAP RPC standard

• **Document**
  - This is a document to be processed – message parts are directly in body
  - Wrapped convention – single message part – looks like RPC style
RPC vs Document

<table>
<thead>
<tr>
<th>WSDL: Message Name: POin</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;part name=prelude type=...&gt;</td>
</tr>
<tr>
<td>&lt;part name=POentries type=...&gt;</td>
</tr>
</tbody>
</table>
| </>

<table>
<thead>
<tr>
<th>WSDL: Operation Name: PurchOrder</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;wsdl:input name=PurchOrderRequest message=\texttt{w:POin}&gt;</td>
</tr>
<tr>
<td>&lt;wsdl:output name=PurchOrderResponse message=\texttt{w:POout}&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Envelope: Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;PurchOrderRequest&gt;</td>
</tr>
<tr>
<td>&lt;Prelude&gt; ... &lt;/&gt;</td>
</tr>
<tr>
<td>&lt;POentries&gt; ... &lt;/&gt;&lt;/&gt;&lt;/&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Envelope: Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;PurchOrderResponse&gt;</td>
</tr>
<tr>
<td>&lt;Result&gt; ... &lt;/&gt;</td>
</tr>
<tr>
<td>&lt;delivSched&gt; ... &lt;/&gt;&lt;/&gt;&lt;/&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Envelope: Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;PurchOrderRequest&gt;</td>
</tr>
<tr>
<td>&lt;Prelude&gt; ... &lt;/&gt;</td>
</tr>
<tr>
<td>&lt;POentries&gt; ... &lt;/&gt;&lt;/&gt;&lt;/&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Envelope: Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;PurchOrderResponse&gt;</td>
</tr>
<tr>
<td>&lt;Result&gt; ... &lt;/&gt;</td>
</tr>
<tr>
<td>&lt;delivSched&gt; ... &lt;/&gt;&lt;/&gt;&lt;/&gt;</td>
</tr>
</tbody>
</table>

RPC Actual messages

Document Actual messages
WSDL

• Defines abstract structure of service interactions
  – Including logical content of messages exchanged

• Defines binding – how the messages are carried and represented
  – Standard binding is for SOAP over HTTP
    ▪ Message is an XML document, with a particular structure
    ▪ Using particular types

• Defines Service as a number of ports, each being address and binding

• A site where you can obtain WSDL definitions of services and see what SOAP messages are produced - http://xmethods.com/
THE END