SOAP MESSAGE EXCHANGE MODEL

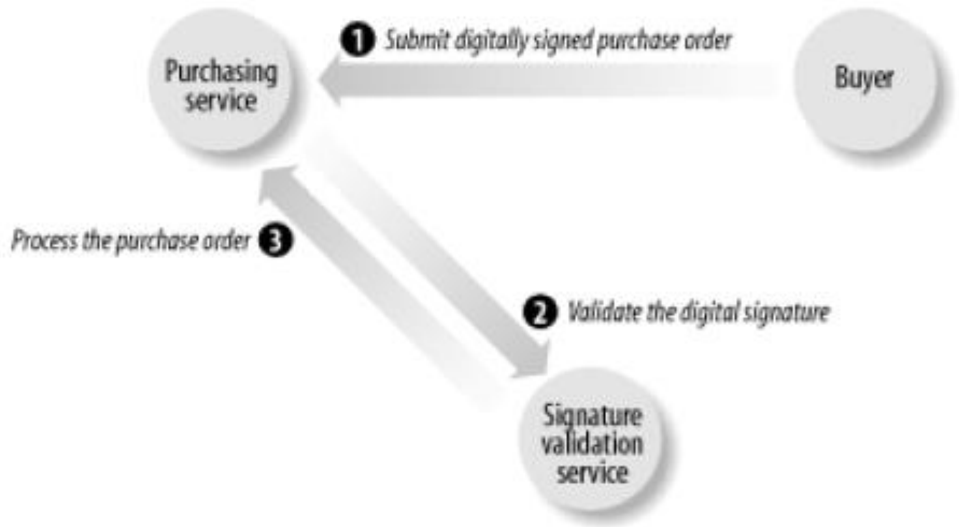
SOAP Message Exchange Model

- Message path & Actors
- Sender-Intermediary1-Intermediary2..-Receiver.

  - . . message path ….   -

  - actor1   - actor 2   -
- Construction of message path is not covered by SOAP specification , there are extensions to SOAP like Microsoft SOAP routing protocol to fill this gap
- WS-Routing (But still there is no standard)
- This mechanism only used in header block not in the body using actor attribute

Example: actor


Purchasing service
Buyer
1 Submit digitally signed purchase order
2 Validate the digital signature
Process the purchase order 3
Signature validation service

```
<s:Envelope xmlns:s="...">
   <s:Header>
      <x:signature actor="uri:SignatureVerifier">
         ...
      </x:signature>
   </s:Header>
   <s:Body>
      <abc:purchaseOrder>...</abc:purchaseOrder>
   </s:Body>
</s:Envelope>
```

WS-Routing Message

```
<s:Envelope xmlns:s="...">
 <s:Header>
  <m:path xmlns:m="http://schemas.xmlsoap.org/rp/"
          s:mustUnderstand="true">
   <m:action>http://www.im.org/chat</m:action>
    <m:to>http://D.com/some/endpoint</m:to>
    <m:fwd>
     <m:via>http://B.com</m:via>
     <m:via>http://C.com</m:via>
    </m:fwd>
    <m:rev>
     <m:via/>
    </m:rev>
    <m:from>mailto:johndoe@acme.com</m:from>
    <m:id>
      uuid:84b9f5d0-33fb-4a81-b02b-5b760641c1d6
    </m:id>
  </m:path>
 </S:Header>
 <S:Body>
   ...
 </S:Body>
</S:Envelope>
```

SOAP-Encoding

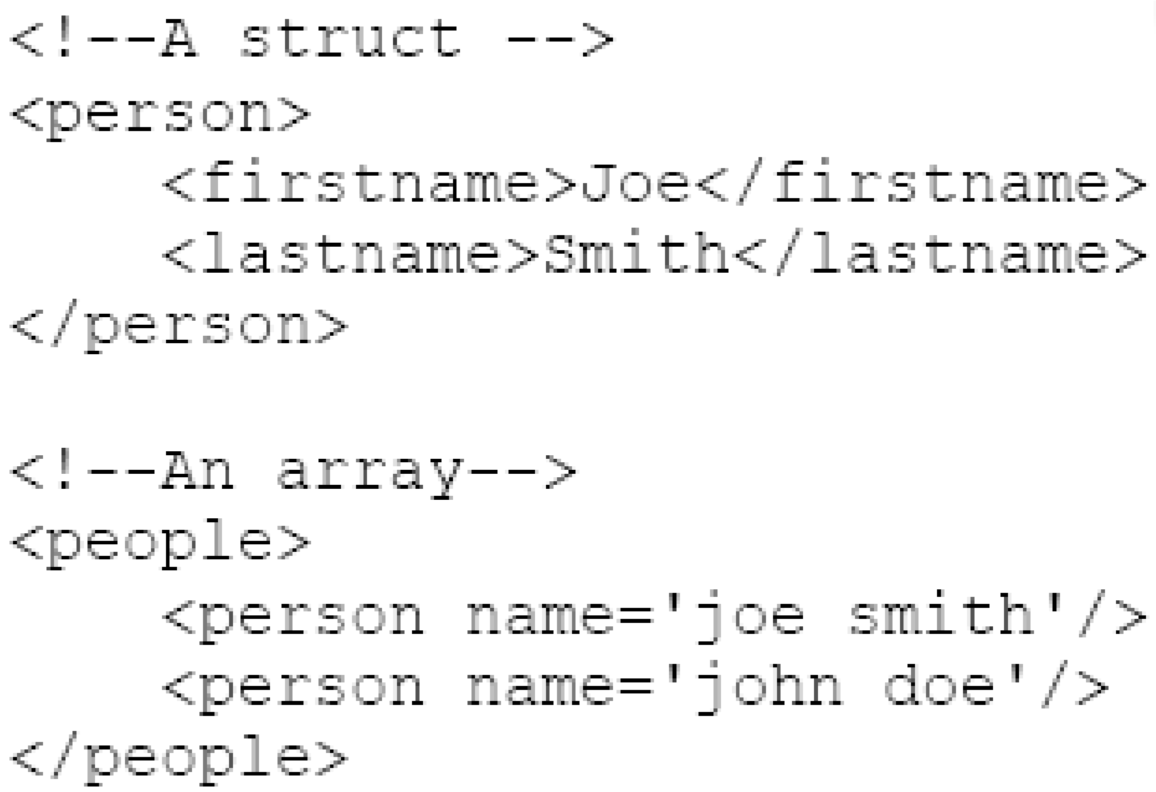
- Envelope – packaging data
- Message – Serialization of data
- Encoding: ‘a simple type system that is the generalization of the common features found in programming languages, databases and semi-structured data’
- Where encoding rules needed?
  - To allow applications dynamically exchange information without a priori knowledge of the types of information to be exchanged.

Encoding…

- Terminology : value & accessor
- value : single / combination of data unit(s)
- accessor:

  element that contains / allows to access a value.
- Accessor Types:
  - Single referenced / multi referenced

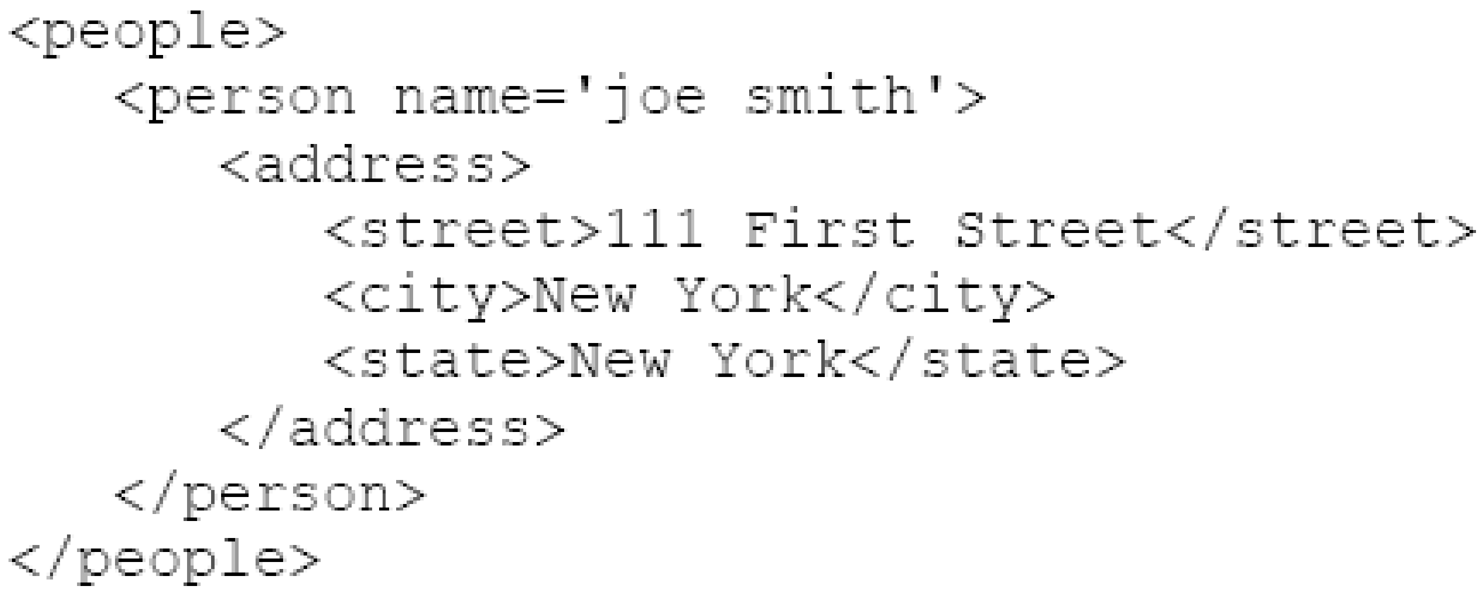
Structure & Arrays



```
<!--A struct -->
<person>
    <firstname>Joe</firstname>
    <lastname>Smith</lastname>
</person>

<!--An array-->
<people>
    <person name='joe smith'/>
    <person name='john doe'/>
</people>
```

Single-referenced accessor



```
<people>
   <person name='joe smith'>
      <address>
         <street>111 First Street</street>
         <city>New York</city>
         <state>New York</state>
      </address>
   </person>
</people>
```

Does not have an identity except as a child of its parent element

Multi-referenced accessor

```
<people>
   <person name='joe smith'>
      <address href='#address-1'
   </person>
   <person name='john doe'>
      <address href='#address-1'
   </person>
</people>
<address id='address-1'>
   <street>111 First Street</street>
   <city>New York</city>
   <state>New York</state>
</address>
```

Uses id to give identity to its value, other accessors can use the href attribute to refer to their values. (the reference can also be an external XML document)

Data Types within SOAP envelope

- SOAP uses 3 difference ways to express the data type of an accessor
  - Use xsi:type on each accessor (data type according to XML schema specification)
    - Ex: <person><name xsi:type="xsd:string">VIT</name></person>
  - Reference an xsd that defines exact data type of a particular element
    - <person xmlns="VIT.xsd"> <name>VIT</name></person>
  - Reference some other type of schema document
    - <person xmlns="urn:vit_namespace"><name>VIT</name></person>

SOAP Data Types

- Anonymous accessor syntax
  - <SOAP-ENC:int> 45 </SOAP-ENC:int>
- Named accessor syntax
  - <value xsi:type =“xsd:int”>45</value>
- Multiple references in XML-encoded data
  - <SOAP-ENC:int>31</SOAP-ENC:int>
  - <SOAP-ENC:int>31</SOAP-ENC:int>

<value xsi:type="xsd:int" id="v1">31</value>

<value href="#v1" />

arrayType attribute

```
<some_array xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="se:string[3]">
   <se:string>Joe</se:string>
   <se:string>John</se:string>
   <se:string>Marsha</se:string>
</some_array>
```

2-dimensional array

```
<data xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[2][]">
   <names href="#names-1"/>
   <names href="#names-2"/>
</data>
<names id="names-1" xsi:type="SOAP-ENC:Array"
       SOAP-ENC:arrayType="xsd:string[2]">
    <name>joe</name>
    <name>john</name>
</names>
<names id="names-2" xsi:type="SOAP-ENC:Array"
       SOAP-ENC:arrayType="xsd:string[2]">
    <name>mike</name>
    <name>bill</name>
</names>
```

Comparison of 2-d and 1-d array

```
<names xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[2,2]">
    <name xsi:type="xsd:string">a1d1</name>
    <name xsi:type="xsd:string">a2d1</name>
    <name xsi:type="xsd:string">a1d2</name>
    <name xsi:type="xsd:string">a2d2</name>
</names>

<names xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[4]">
    <name xsi:type="xsd:string">a1d1</name>
    <name xsi:type="xsd:string">a2d1</name>
    <name xsi:type="xsd:string">a3d1</name>
    <name xsi:type="xsd:string">a4d1</name>
</names>
```

SOAP-ENC:offset for partially transmitted arrays

```
<names xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[5]"
       SOAP-ENC:offset="[2]">
    <name>Item 4</name>
    <name>Item 5</name>
</names>
```

SOAP-Serialization of sparse arrays

```
<names xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[10,10]">
    <name SOAP-ENC:position="[2,5]">data</name>
    <name SOAP-ENC:position="[5,2]">data</name>
</names>
```

Null accessors

```
<name xsi:type="xsd:string" xsi:nil="true" />
```

SOAP Web Services (Refer Lecture Notes)

- Creating web services in .NET
- Deployment
- Invoking the service using SOAP
- System.Web.Services.WebServiceBindingAttribute
- System.Web.Services.Protocols.SoapHttpClientProtocol
- Interoperability issues
  - SOAP faults,SOAP::LITE, Perl client to work with .NET.