

Quick Intro to XML Schemas & SOAP

Noah Mendelsohn
Lotus Development Corp.
Oct. 12, 2000

How SOAP uses Schemas

- To define SOAP's XML vocabulary
- Optionally: to define your msg. vocabulary
- SOAP encoding uses schema "types"
 - ▶ Schema builtin datatypes: integer, float, date, etc.
 - ▶ SOAP builtins for arrays & structs
 - ▶ Types you define in schema
 - ▶ You can indicate types:
 - Directly in your message using `xsi:type`
 - Optionally: in an external schema document

Schema WG Status

- Hope for candidate recommendation soon
- Three documents
 - ▶ Structures: the overall language
 - ▶ Datatypes: integer, float, date, etc.
 - ▶ Primer: start with this!
- Public working drafts available

A Schema

```
<xsd:schema xmlns:xsd="..." targetNamespace="yourURI">
  <xsd:element name="purchaseOrder" type="POType"/>
  <xsd:element name="comment" type="xsd:string"/>

  <xsd:complexType name="POType">
    <xsd:sequence>
      <xsd:element name="shipTo" type="USAddress"/>
      <xsd:element name="billTo" type="USAddress"/>
      <xsd:element ref="comment" minOccurs="0"/>
      <xsd:element name="items" type="Items"/>
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>
  .....
</xsd:schema>
```

Namespaces and Schema Documents

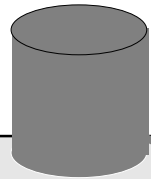
A SOAP Message

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://{soaporg}/envelope/"
  SOAP-ENV:encodingStyle=
    "http://{soaporg}/encoding/">

  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

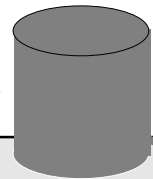
SOAP Schema



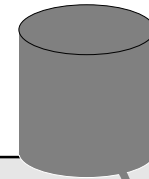
soapenv.xsd: *validates SOAP's vocabulary*

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://{soaporg}/envelope/"
  SOAP-ENV:encodingStyle=
    "http://{soaporg}/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


SOAP Schema & User Schema



soapenv.xsd



stockquote.xsd

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://{soaporg}/envelope/"
  SOAP-ENV:encodingStyle=
    "http://{soaporg}/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Declaring Elements and Attributes

A Sample Instance

```
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <billTo country="US">...</billTo>
  <comment>Hurry, my lawn is going wild!</comment>
  <items>
    <item partNum="872-AA">...</item>
    <item partNum="926-AA">...</item>
  </items>
</purchaseOrder>
```

A Schema

```
<xsd:schema xmlns:xsd="..." targetNamespace="yourURI">
  <xsd:element name="purchaseOrder" type="POType"/>
  <xsd:element name="comment" type="xsd:string"/>

  <xsd:complexType name="POType">
    <xsd:sequence>
      <xsd:element name="shipTo" type="USAddress"/>
      <xsd:element name="billTo" type="USAddress"/>
      <xsd:element ref="comment" minOccurs="0"/>
      <xsd:element name="items" type="Items"/>
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>
  .....
</xsd:schema>
```

Elements have Types

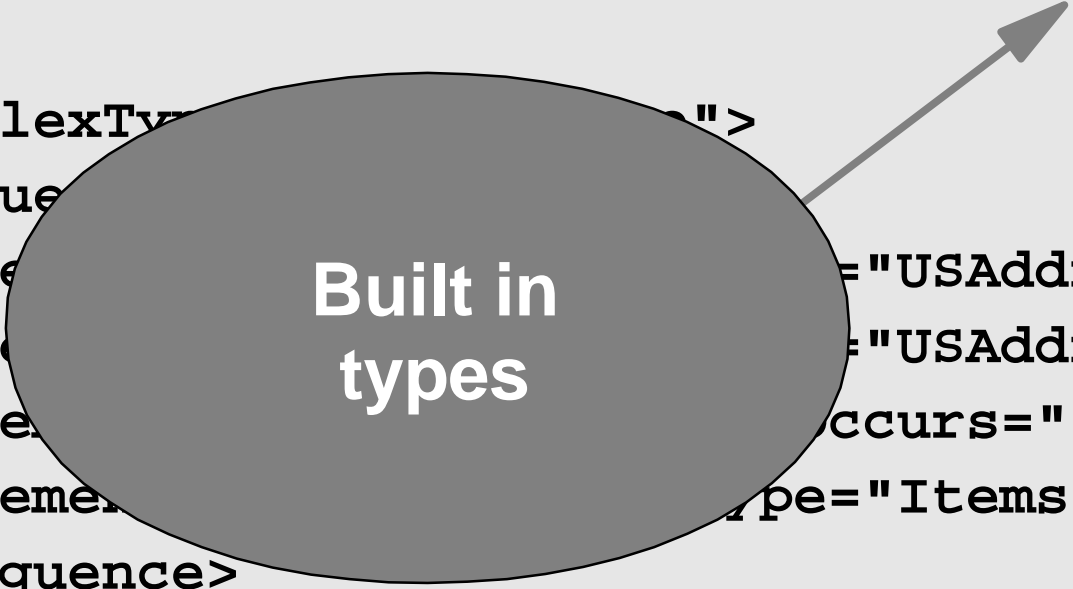
```
<xsd:schema xmlns:xsd="..." targetNamespace="yourURI">
  <xsd:element name="purchaseOrder" type="POType"/>
  <xsd:element name="comment" type="xsd:string"/>

  <xsd:complexType name="POType">
    <xsd:sequence>
      <xsd:element name="shipTo" type="USAddress"/>
      <xsd:element name="billTo" type="USAddress"/>
      <xsd:element ref="comment" minOccurs="0"/>
      <xsd:element name="items" type="Items"/>
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>
  . . . . .
</xsd:schema>
```

Elements have Types

```
<xsd:schema xmlns:xsd="..." targetNamespace="yourURI">
  <xsd:element name="purchaseOrder" type="POType"/>
  <xsd:element name="comment" type="xsd:string"/>

  <xsd:complexType name="POType">
    <xsd:sequence base="xsd:string">
      <xsd:element name="address" type="USAddress"/>
      <xsd:element name="city" type="USAddress"/>
      <xsd:element name="state" type="USAddress" minOccurs="0"/>
      <xsd:element name="items" type="Items"/>
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>
  .....
</xsd:schema>
```

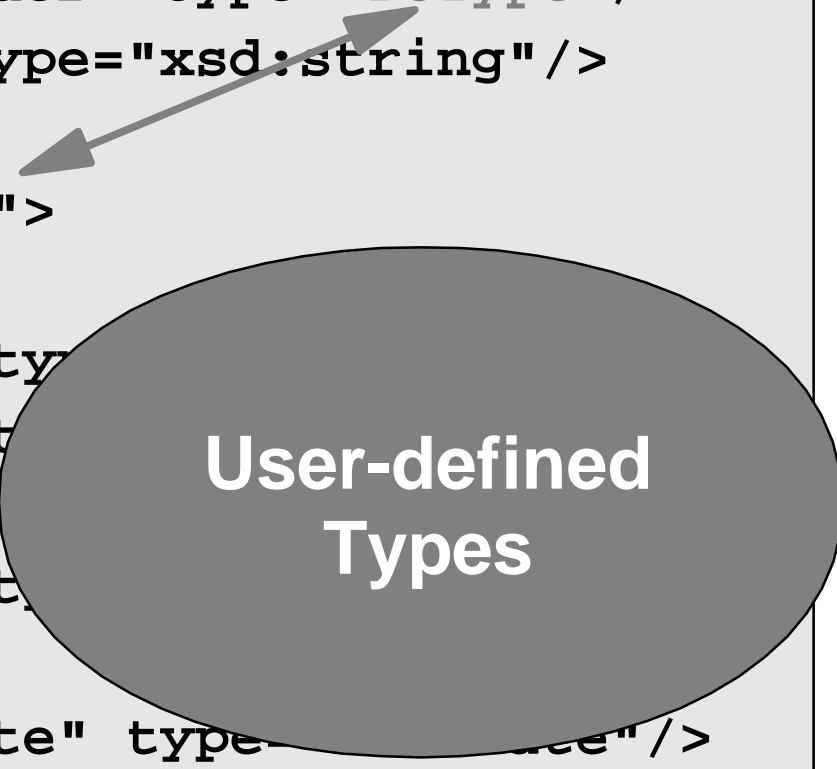


Built in types

Elements have Types

```
<xsd:schema xmlns:xsd="..." targetNamespace="yourURI">
  <xsd:element name="purchaseOrder" type="POType"/>
  <xsd:element name="comment" type="xsd:string"/>

  <xsd:complexType name="POType">
    <xsd:sequence>
      <xsd:element name="shipTo" type="..." />
      <xsd:element name="billTo" type="..." />
      <xsd:element ref="comment" />
      <xsd:element name="items" type="..." />
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>
  .....
</xsd:schema>
```

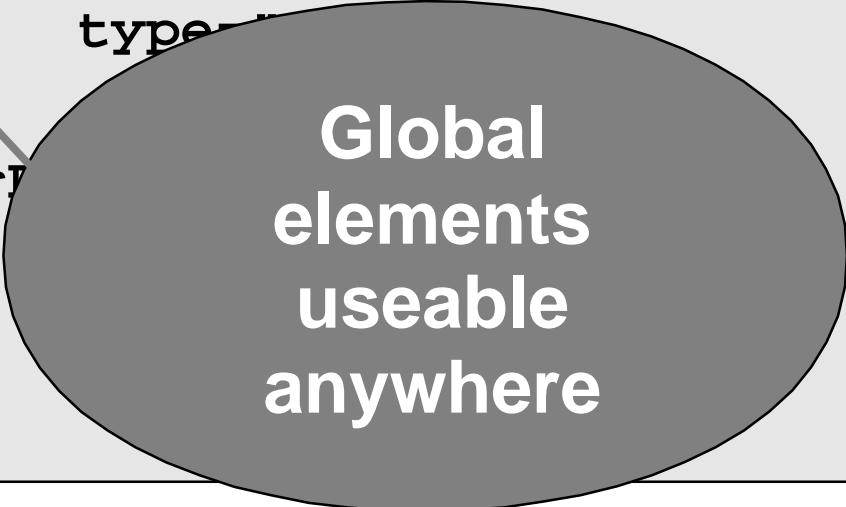


The diagram illustrates the relationship between an XML element and its type. An arrow points from the `<xsd:element name="purchaseOrder" type="POType"/>` line to the `<xsd:complexType name="POType">` definition. A grey oval callout box labeled "User-defined Types" is positioned over the `<xsd:complexType name="POType">` definition, indicating that this is a user-defined type.

Global elements

```
<xsd:schema xmlns:xsd="..." targetNamespace="yourURI">
  <xsd:element name="purchaseOrder" type="POType"/>
  <xsd:element name="comment" type="xsd:string"/>

  <xsd:complexType name="POType">
    <xsd:sequence>
      <xsd:element name="shipTo" type="USAddress"/>
      <xsd:element name="billTo" type="USAddress"/>
      <xsd:element ref="comment" minOccurs="0"/>
      <xsd:element name="items" type="..." />
    </xsd:sequence>
    <xsd:attribute name="orderId" type="xsd:string"/>
  </xsd:complexType>
  .....
</xsd:schema>
```



Global elements useable anywhere

The diagram consists of a grey oval containing the text 'Global elements useable anywhere'. Two arrows originate from this oval: one points to the '<xsd:element ref="comment" minOccurs="0"/>' line in the code, and the other points to the '<xsd:element name="comment" type="xsd:string"/>' line in the code, illustrating that the global element 'comment' is referenced from within a complex type definition.

Locally scoped elements

```
<xsd:schema xmlns:xsd="..." target="...">
  <xsd:element name="purchaseOrder" type="PurchaseOrder" />
  <xsd:element name="comment" type="Text" />

  <xsd:complexType name="POType" base="PurchaseOrder">
    <xsd:sequence>
      <xsd:element name="shipTo" type="USAddress" />
      <xsd:element name="billTo" type="USAddress" />
      <xsd:element ref="comment" minOccurs="0" />
      <xsd:element name="items" type="Items" />
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date" />
  </xsd:complexType>
  . . . . .
</xsd:schema>
```

Locally
scoped
elements
useable only
in parent
type

Attributes have Types

```
<xsd:schema xmlns:xsd="..." targetNamespace="yourURI">
  <xsd:element name="purchaseOrder" type="POType"/>
  <xsd:element name="comment" type="xsd:string"/>

  <xsd:complexType name="POType">
    <xsd:sequence>
      <xsd:element name="shipTo" type="USAddress"/>
      <xsd:element name="billTo" type="USAddress"/>
      <xsd:element ref="comment" minOccurs="0"/>
      <xsd:element name="items" type="Items"/>
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>
  .....
</xsd:schema>
```

Simple and Complex Types

Simple types

Attributes

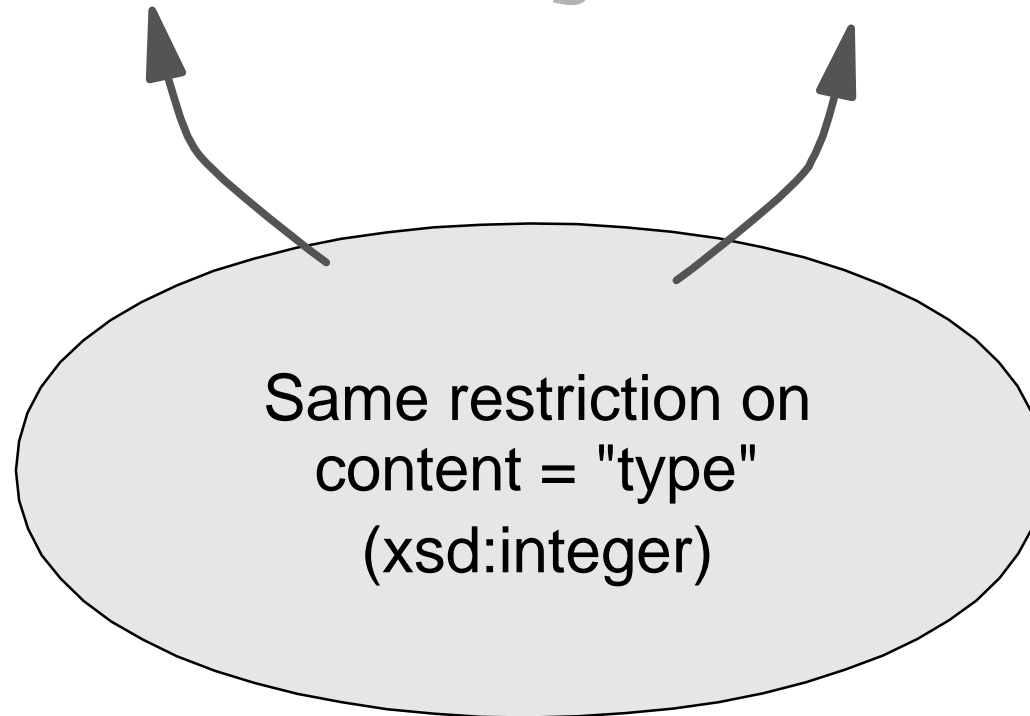
`width="10" height="20"`

Simple types

Attributes

`width="10"`

`height="20"`



Simple types

Attributes

`width="10"`

`height="20"`

Elements

`<width>`

`10`

`</width>`

`<height>`

`20`

`</height>`

Simple type captures what's common:

- It's an integer
- Type has a name (`xsd:integer`)
- Simple types work on attrs and elements

Types for Elements

Simple type

```
<width>  
  10  
</width>
```

```
<height>  
  20  
</height>
```

Complex type

```
<width unit="cm">  
  10  
</width>
```

```
<height unit="cm">  
  20  
</height>
```

Type(s) capture what's common:

- It's an integer + attribute
- Type has a name (yourns:measure)

More Complex Types for Elements

Element

```
<ShipTo>  
  <street>1 Main St.</street>  
  <city state="NY">Albany</city>  
</ShipTo>  
  
<BillTo>  
  <street>13 Market St.</street>  
  <city state="CA">San Jose</city>  
</BillTo>
```

Complex types are for elements:

- Sharing content models, attr lists.
- Type has a name (somens:address)

Self-describing SOAP Msgs

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://{soaporg}/envelope/"
  SOAP-ENV:encodingStyle=
    "http://{soaporg}/encoding/">

  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol xsi:type="xsd:string">
        DIS
      </symbol>
      <quoteDate xsi:type="xsd:date">
        1999-05-21
      </quoteDate>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Summary:

How SOAP uses Schemas

- To define SOAP's XML vocabulary
- Optionally: to define your msg. vocabulary
- SOAP encoding uses schema "types"
 - ▶ Schema builtin datatypes: integer, float, date, etc.
 - ▶ SOAP builtins for arrays & structs
 - ▶ Types you define in schema
 - ▶ You can indicate types:
 - Directly in your message using xsi:type
 - Optionally: in an external schema document