

[3] E. Whitehead, M. Murata, "XML Media Types", RFC2376, UC Irvine, Fuji Xerox Info. Systems, July 1998

[4] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

[5] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, U.C. Irvine, DEC W3C/MIT, DEC, W3C/MIT, W3C/MIT, January 1997

[6] H. Nielsen, P. Leach, S. Lawrence, "An HTTP Extension Framework", RFC 2774, Microsoft, Microsoft, Agranat Systems

[7] W3C Recommendation "The XML Specification"

[8] W3C Recommendation "Namespaces in XML"

[9] W3C Working Draft "XML Linking Language". This is work in progress.

[10] W3C Working Draft "XML Schema Part 1: Structures". This is work in progress.

[11] W3C Working Draft "XML Schema Part 2: Datatypes". This is work in progress.

[12] Transfer Syntax NDR, in "DCE 1.1: Remote Procedure Call"

[13] N. Freed, N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC2045, Innosoft, First Virtual, November 1996

A. SOAP Envelope Examples

A.1 Sample Encoding of Call Requests

Example 5 Similar to Example 1 but with a Mandatory Header

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"
```

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
  <SOAP-ENV:Header>
    <t:Transaction
      xmlns:t="some-URI"
      SOAP-ENV:mustUnderstand="1" />
    5
  </t:Transaction>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
  <m:GetLastTradePrice xmlns:m="Some-URI">
    <symbol>DEF</symbol>
  </m:GetLastTradePrice>
```

The members of a Compound Value are encoded as accessor elements. When accessors are distinguished by their name (as for example in a struct), the accessor name is used as the element name. Accessors whose names are local to their containing types have unqualified element names; all others have qualified names.

The following is an example of a struct of type "Book":

```
<e:Book>
  <author>Henry Ford</author>
  <preface>Prefatory text</preface>
  <intro>This is a book.</intro>
</e:Book>
```

And this is a schema fragment describing the above structure:

```
<element name="Book">
  <complexType>
    <element name="author" type="xsd:string"/>
    <element name="preface" type="xsd:string"/>
    <element name="intro" type="xsd:string"/>
  </complexType>
</e:Book>
```

Below is an example of a type with both simple and complex members. It shows two levels of referencing. Note that the "href" attribute of the "Author" accessor element is a reference to the value whose "id" attribute matches. A similar construction appears for the "Address".

```
<e:Book>
  <title>My Life and Work</title>
  <author href="#Person-1"/>
</e:Book>
<e:Person id="Person-1">
  <name>Henry Ford</name>
  <address href="#Address-2"/>
</e:Person>
<e:Address id="Address-2">
  <email>mailto:henryford@hotmail.com</email>
  <web>http://www.henryford.com</web>
</e:Address>
```

The form above is appropriate when the "Person" value and the "Address" value are multi-reference. If these were instead both single-reference, they SHOULD be embedded, as follows:

```
<e:Book>
  <title>My Life and Work</title>
  <author>
    <name>Henry Ford</name>
    <address>
      <email>mailto:henryford@hotmail.com</email>
      <web>http://www.henryford.com</web>
    </address>
  </author>
</e:Book>
```

If instead there existed a restriction that no two persons can have the same address in a given instance and that an address can be either a Street-address or an Electronic-address, a Book with two authors would be encoded as follows:

```
<e:Book>
  <title>My Life and Work</title>
  <firstauthor href="#Person-1" />
  <secondauthor href="#Person-2"/>
</e:Book>
<e:Person id="Person-1">
  <name>Henry Ford</name>
  <address xsi:type="m:Electronic-address">
    <email>mailto:henryford@hotmail.com</email>
    <web>http://www.henryford.com</web>
  </address>
</e:Person>
<e:Person id="Person-2">
  <name>Samuel Crowther</name>
  <address xsi:type="n:Street-address">
    <street>Martin Luther King Rd</street>
    <city>Raleigh</city>
    <state>North Carolina</state>
  </address>
</e:Person>
```

Serializations can contain references to values not in the same resource:

```
<e:Book>
  <title>Paradise Lost</title>
  <firstauthor href="http://www.dartmouth.edu/~milton"/>
</e:Book>
```

And this is a schema fragment describing the above structures:

```
<element name="Book" type="tns:Book"/>
<complexType name="Book">
  <!-- Either the following group must occur or else the
  href attribute must appear, but not both. -->
  <sequence minOccurs="0" maxOccurs="1">
    <element name="title" type="xsd:string"/>
    <element name="firstauthor" type="tns:Person"/>
    <element name="secondauthor" type="tns:Person"/>
  </sequence>
  <attribute name="href" type="uriReference"/>
  <attribute name="id" type="ID"/>
  <anyAttribute namespace="##other"/>
</complexType>

<element name="Person" base="tns:Person"/>
<complexType name="Person">
  <!-- Either the following group must occur or else the
  href attribute must appear, but not both. -->
  <sequence minOccurs="0" maxOccurs="1">
    <element name="name" type="xsd:string"/>
  </sequence>
  <attribute name="id" type="ID"/>
  <attribute name="href" type="uriReference"/>
  <anyAttribute namespace="##other"/>
</complexType>
```

```

<element name="address" type="tns:Address"/>
</sequence>
<attribute name="href" type="uriReference"/>
<attribute name="id" type="ID"/>
<anyAttribute namespace="##other"/>
</complexType>

```

```

<element name="Address" base="tns:Address"/>
<complexType name="Address">
  <!-- Either the following group must occur or else the
       href attribute must appear, but not both. -->
  <sequence minOccurs="0" maxOccurs="1">
    <element name="street" type="xsd:string"/>
    <element name="city" type="xsd:string"/>
    <element name="state" type="xsd:string"/>
  </sequence>
  <attribute name="href" type="uriReference"/>
  <attribute name="id" type="ID"/>
  <anyAttribute namespace="##other"/>
</complexType>

```

5.4.2 Arrays

SOAP arrays are defined as having a type of "SOAP-ENC:Array" or a type derived there from (see also [rule 8](#)). Arrays are represented as element values, with no specific constraint on the name of the containing element (just as values generally do not constrain the name of their containing element).

Arrays can contain elements which themselves can be of any type, including nested arrays. New types formed by restrictions of SOAP-ENC:Array can also be created to represent, for example, arrays limited to integers or arrays of some user-defined enumeration.

The representation of the value of an array is an ordered sequence of elements constituting the items of the array. Within an array value, element names are not significant for distinguishing accessors. Elements may have any name. In practice, elements will frequently be named so that their declaration in a schema suggests or determines their type. As with compound types generally, if the value of an item in the array is a single-reference value, the item contains its value. Otherwise, the item references its value via an "href" attribute.

The following example is a schema fragment and an array containing integer array members.

```

<element name="myFavoriteNumbers"
  type="SOAP-ENC:Array"/>

<myFavoriteNumbers
  SOAP-ENC:arrayType="xsd:int[2]">
  <number>3</number>
  <number>4</number>
</myFavoriteNumbers>

```

In that example, the array "myFavoriteNumbers" contains several members each of which

Array values may be structs or other compound values. For example an array of "xyz:Order" structs :

```
<SOAP-ENC:Array SOAP-ENC:arrayType="xyz:Order[2]">
  <Order>
    <Product>Apple</Product>
    <Price>1.56</Price>
  </Order>
  <Order>
    <Product>Peach</Product>
    <Price>1.48</Price>
  </Order>
</SOAP-ENC:Array>
```

Arrays may have other arrays as member values. The following is an example of an array of two arrays, each of which is an array of strings.

```
<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:string[][2]">
  <item href="#array-1"/>
  <item href="#array-2"/>
</SOAP-ENC:Array>
<SOAP-ENC:Array id="array-1" SOAP-ENC:arrayType="xsd:string[2]">
  <item>r1c1</item>
  <item>r1c2</item>
  <item>r1c3</item>
</SOAP-ENC:Array>
<SOAP-ENC:Array id="array-2" SOAP-ENC:arrayType="xsd:string[2]">
  <item>r2c1</item>
  <item>r2c2</item>
</SOAP-ENC:Array>
```

The element containing an array value does not need to be named "SOAP-ENC:Array". It may have any name, provided that the type of the element is either SOAP-ENC:Array or is derived from SOAP-ENC:Array by restriction. For example, the following is a fragment of a schema and a conforming instance array.

```
<simpleType name="phoneNumber" base="string"/>

<element name="ArrayOfPhoneNumbers">
  <complexType base="SOAP-ENC:Array">
    <element name="phoneNumber" type="ins:phoneNumber"
maxOccurs="unbounded"/>
  </complexType>
  <anyAttribute/>
</element>

<xyz:ArrayOfPhoneNumbers SOAP-ENC:arrayType="xyz:phoneNumber[2]">
  <phoneNumber>206-555-1212</phoneNumber>
  <phoneNumber>1-888-123-4567</phoneNumber>
</xyz:ArrayOfPhoneNumbers>
```

Arrays may be multi-dimensional. In this case, more than one size will appear within the size part of the arrayType attribute:

```
<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:string[2,3]">
```

```

<item>r1c1</item>
<item>r1c2</item>
<item>r1c3</item>
<item>r2c1</item>
<item>r2c2</item>
<item>r2c3</item>
</SOAP-ENC:Array>
    
```

While the examples above have shown arrays encoded as independent elements, array values MAY also appear embedded and SHOULD do so when they are known to be single reference.

The following is an example of a schema fragment and an array of phone numbers embedded in a struct of type "Person" and accessed through the accessor "phone-numbers":

```

<simpleType name="phoneNumber" base="string"/>

<element name="ArrayOfPhoneNumbers">
  <complexType base="SOAP-ENC:Array">
    <element name="phoneNumber" type="tns:phoneNumber"
      maxOccurs="unbounded"/>
  </complexType>
</anyAttribute/>
</element>

<element name="Person">
  <complexType>
    <element name="name" type="string"/>
    <element name="phoneNumbers" type="tns:ArrayOfPhoneNumbers"/>
  </complexType>
</element>

<xyz:Person>
  <name>John Hancock</name>
  <phoneNumbers SOAP-ENC:arrayType="xyz:phoneNumber[2]">
    <phoneNumber>206-555-1212</phoneNumber>
    <phoneNumber>1-888-123-4567</phoneNumber>
  </phoneNumbers>
</xyz:Person>
    
```

Here is another example of a single-reference array value encoded as an embedded element whose containing element name is the accessor name:

```

<xyz:PurchaseOrder>
  <CustomerName>Henry Ford</CustomerName>
  <ShipTo>
    <Street>5th Ave</Street>
    <City>New York</City>
    <State>NY</State>
    <Zip>10010</Zip>
  </ShipTo>
  <PurchaseLineItems SOAP-ENC:arrayType="Order[2]">
    <Order>
      <Product>Apple</Product>
    
```

```

<Price>1.56</Price>
</Order>
<Order>
  <Product>Peach</Product>
  <Price>1.48</Price>
</Order>
</PurchaseLineItems>
</xyz:PurchaseOrder>

```

5.4.2.1 Partially Transmitted Arrays

SOAP provides support for partially transmitted arrays, known as "varying" arrays in some contexts [12]. A partially transmitted array indicates in an "SOAP-ENC:offset" attribute the zero-origin offset of the first element transmitted. If omitted, the offset is taken as zero.

The following is an example of an array of size five that transmits only the third and fourth element counting from zero:

```

<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:string[5]" SOAP-ENC:offset="[2]">
  <item>The third element</item>
  <item>The fourth element</item>
</SOAP-ENC:Array>

```

5.4.2.2 Sparse Arrays

SOAP provides support for sparse arrays. Each element representing a member value contains a "SOAP-ENC:position" attribute that indicates its position within the array. The following is an example of a sparse array of two-dimensional arrays of strings. The size is 4 but only position 2 is used:

```

<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:string[,] [4]">
  <SOAP-ENC:Array href="#array-1" SOAP-ENC:position="[2]" />
</SOAP-ENC:Array>
<SOAP-ENC:Array id="array-1" SOAP-ENC:arrayType="xsd:string[10,10]">
  <item SOAP-ENC:position="[2,2]">Third row, third col</item>
  <item SOAP-ENC:position="[7,2]">Eighth row, third col</item>
</SOAP-ENC:Array>

```

If the only reference to array-1 occurs in the enclosing array, this example could also have been encoded as follows:

```

<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:string[,] [4]">
  <SOAP-ENC:Array SOAP-ENC:position="[2]"
SOAP-ENC:arrayType="xsd:string[10,10]">
  <item SOAP-ENC:position="[2,2]">Third row, third col</item>
  <item SOAP-ENC:position="[7,2]">Eighth row, third col</item>
</SOAP-ENC:Array>
</SOAP-ENC:Array>

```

5.4.3 Generic Compound Types

The encoding rules just cited are not limited to those cases where the accessor names are known in advance. If accessor names are known only by inspection of the immediate values to be encoded, the same rules apply, namely that the accessor is encoded as an

```
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 6 Similar to Example 1 but with multiple request parameters

```
POST /StockQuote HTTP/1.1
Host: www.stockquotesever.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"
```

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePriceDetailed
      xmlns:m="Some-URI">
      <Symbol>DEF</Symbol>
      <Company>DEF Corp</Company> } multi
      <Price>34.1</Price>
    </m:GetLastTradePriceDetailed>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A.2 Sample Encoding of Response

Example 7 Similar to Example 2 but with a Mandatory Header

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
```

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Header>
    <t:Transaction
      xmlns:t="some-URI"
      xsi:type="xsd:int" mustUnderstand="1">
      5
    </t:Transaction>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <m:GetLastTradePriceResponse
      xmlns:m="Some-URI">
      <Price>34.5</Price>
    </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example 8 Similar to Example 2 but with a Struct

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
```

Content-Length: nnnn

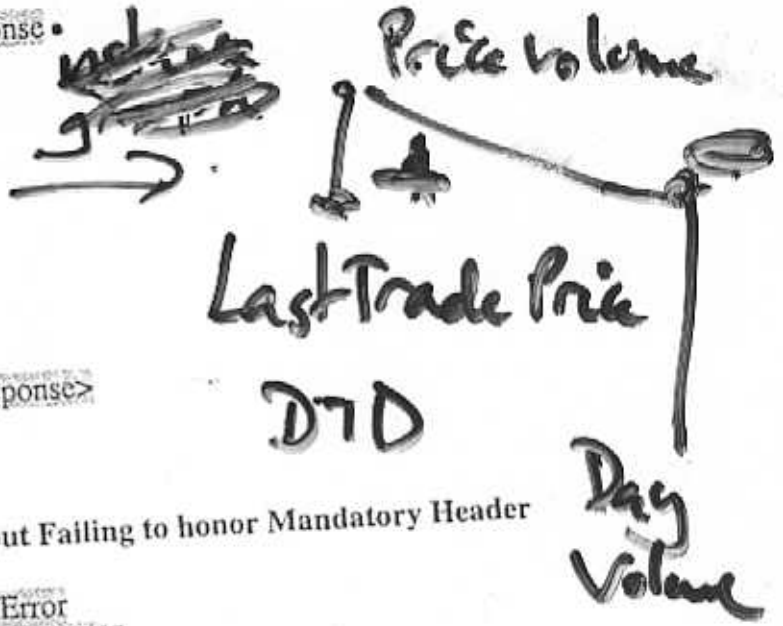
Get Last Trade Price Response
(34.5, 10000)

```

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePriceResponse
      xmlns:m="Some-URI">
      <PriceAndVolume>
        <LastTradePrice>
          34.5
        </LastTradePrice>
        <DayVolume>
          10000
        </DayVolume>
      </PriceAndVolume>
    </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

34.5 10000



Example 9 Similar to Example 2 but Failing to honor Mandatory Header

HTTP/1.1 500 Internal Server Error
 Content-Type: text/xml; charset="utf-8"
 Content-Length: nnnn

```

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:MustUnderstand</faultcode>
      <faultstring>SOAP Must Understand Error</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Example 10 Similar to Example 2 but Failing to handle Body

HTTP/1.1 500 Internal Server Error
 Content-Type: text/xml; charset="utf-8"
 Content-Length: nnnn

```

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Server</faultcode>
      <faultstring>Server Error</faultstring>
      <detail>
        <e:myfaultdetails xmlns:e="Some-URI">
          <message>
            My application didn't work
          </message>
          <errorcode>
            1001
          </errorcode>
        </e:myfaultdetails>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

```
</errorcode>  
</e:myfaultdetails>  
</detail>  
</SOAP-ENV:Fault>  
</SOAP-ENV:Body>  
</SOAP-ENV:Envelope>
```
