Table of Contents

Java and SOAP By Robert Englander ISBN: 0-596-00175-4 Publisher: O'Reilly
Print Publication Date: 2002/05/01

Chapter 3. SOAP Data Encoding

When sending data over a network, the data must comply with the underlying transmission protocol, and be formatted in such a way that both the sending and receiving parties understand its meaning. This is what we refer to as *data encoding*. Data encoding encompasses the organization of the data structure, the type of data transferred, and of course the data's value. Just like in Java, it is the data that gets serialized, not the behavior. Data encoding and serialization rules help the parties involved in a SOAP transaction to understand the meaning and content of the message. The model for SOAP encoding is based on XML data encoding, but the encoding constrains or alters those rules to fit the intended purpose of SOAP. I think you'll find that the data requirements of most systems can be easily represented using the encoding rules presented here.

3.1. Schemas and Namespaces

Namespaces provide the mechanism used to determine how element and attribute names are interpreted. Because XML allows arbitrary element names, it needs a mechanism for specifying which dictionary should be used to look up the meaning of any given name. The encoding style defined in section 5 of the SOAP specification is the most commonly used encoding style in SOAP. This encoding style, which is defined in the schema referenced by the `http://schemas.xmlsoap.org/encoding` namespace, is often referred to as "SOAP Section 5." In SOAP messages, this namespace is by convention referenced using a namespace qualifier such as `SOAP-ENC` or `soapenc`. In our examples we will use the `SOAP-ENC` namespace qualifier to refer to this namespace.

It's important to understand that, in SOAP, schemas are used as references to definitions of data elements. They aren't used to validate SOAP message data in standard SOAP processing, although there's nothing stopping you from doing that on your own. References to schemas are often used as namespaces in order to qualify a serialized data element. It's up to the developer or the underlying framework to understand the structure or meaning of the data and to code to it accordingly.

3.2. Serialization Rules

The XML elements in a SOAP message are either *independent* or *embedded*. Because of the hierarchical nature of XML, most elements are embedded as subelements of other elements. Independent elements, then, are not subelements of any other elements; they appear at the top level of a serialization.

Java and SOAP By Robert Englander ISBN: 0-596-00175-4 Publisher: O'Reilly
Print Publication Date: 2002/05/01

All of the values in a SOAP message are encoded as the content of an element. Data values cannot appear by themselves outside the confines of an element. That does not mean, however, that every XML element contains a value. For instance, compound data types like structs or arrays contain subelements that contain the actual data values. The elements that define these compound data values do not contain the data directly. Compound types will be covered a little later.

3.3. Indicating Type

Every element that contains a value must also indicate the type of the data. There are a few different ways to indicate the data type. The first mechanism is to include an `xsi:type` attribute as part of the element. This means that the attribute is named `type`, as defined by the namespace indicated by `xsi`. The value assigned to `xsi:type` must be a valid type identifier such as `xsd:float`, `xsd:string`, etc. In the second mechanism, the value can be an element of an array that already constrains the type of its constituent parts to a particular data type. In this case, no explicit type declaration for the individual values is necessary; we'll see this later when we talk about arrays. Finally, the element name itself can be related to some type that can be determined by looking at the associated XML schema. The following extract from an XML schema defines a compound data type:

```
<element name="Automobile" type="Automobile"/>
<complexType name="Automobile">
     <element name="make" type="xsd:string"/>
     <element name="model" type="xsd:string"/>
     <element name="year" type="xsd:int"/>
</complexType>
```

3.4. Default Values

The SOAP specification talks briefly about the handling of default values. It states that the omission of an element from the transmitted data can imply that a default value is to be used. The C++ language provides a good programming analogy to this topic, since it allows you to define default values for method parameters, thus allowing the caller to omit values for those parameters. Here's what just such a method definition in C++ looks like:

```
int getValue (int param1, int param2 = 1);
```

3.5. The SOAP Root Attribute

Linked lists, trees, and directed graphs are common data structures. However, we've yet to see a serialization mechanism that identifies where the starting point, or root, of such a structure might be. The SOAP root attribute is used for just this purpose. Assigning the value of 1 to the `SOAP-ENC:root` attribute of an element identifies it as the root of a structure such as a linked list.

Here is some Java code for the nodes of a singly linked list of integers, followed by an instance of a list containing three nodes. The nodes contain the values 50, 60, and 70, and

Java and SOAP By Robert Englander ISBN: 0-596-00175-4 Publisher: O'Reilly
Print Publication Date: 2002/05/01

they are placed into the list in ascending order with the node containing the value 50 being the head (or root) of the list.

Java and SOAP By Robert Englander ISBN: 0-596-00175-4 Publisher: O'Reilly
Print Publication Date: 2002/05/01