XML-RPC - DATA MODEL

The XML-RPC specification defines six basic data types and two compound data types that represent combinations of types.

Basic Data Types in XML-RPC

Type	Value	Examples
int or i4	32-bit integers between -2,147,483,648 and 2,147,483,647.	<int>27</int> <i4>27</i4>
double	64-bit floating-point numbers	<double>27.31415</double> <double>-1.1465</double>
Boolean	true 1 or false 0	<boolean>1</boolean> <boolean>0</boolean>
string	ASCII text, though many implementations support Unicode	<string>Hello</string> <string>bonkers! @</string>
dateTime.iso8601	Dates in ISO8601 format: CCYYMMDDTHH:MM:SS	<dateTime.iso8601> 20021125T02:20:04 </dateTime.iso8601> <dateTime.iso8601> 20020104T17:27:30 </dateTime.iso8601>
base64	Binary information encoded as Base 64, as defined in RFC 2045	<base64>SGVsbG8sIFdvcmxkIQ==</base64>

These basic types are always enclosed in *value* elements. Strings *andonlystrings* may be enclosed in a *value* element but omit the *string* element. These basic types may be combined into two more complex types, arrays, and structs. Arrays represent sequential information, while structs represent name-value pairs, much like hashtables, associative arrays, or properties.

Arrays are indicated by the *array* element, which contains a *data* element holding the list of values. Like other data types, the *array* element must be enclosed in a *value* element. For example, the following *array*contains four strings:

```
<value>
   <array>
```

```
      <data>
         <value><string>This </string></value>
         <value><string>is </string></value>
         <value><string>an </string></value>
         <value><string>array.</string></value>
      </data>
   </array>
</value>
```

The following array contains four integers:

```
<value>
   <array>
      <data>
         <value><int>7</int></value>
         <value><int>1247</int></value>
         <value><int>-91</int></value>
         <value><int>42</int></value>
      </data>
   </array>
</value>
```

Arrays can also contain mixtures of different types, as shown here:

```
<value>
   <array>
      <data>
         <value><boolean>1</boolean></value>
         <value><string>Chaotic collection, eh?</string></value>
         <value><int>-91</int></value>
         <value><double>42.14159265</double></value>
      </data>
   </array>
</value>
```

Creating multidimensional arrays is simple - just add an array inside of an array:

```
<value>
   <array>
      <data>

         <value>
            <array>
               <data>
                  <value><int>10</int></value>
                  <value><int>20</int></value>
                  <value><int>30</int></value>
               </data>
            </array>
         </value>

         <value>
            <array>
               <data>
                  <value><int>15</int></value>
                  <value><int>25</int></value>
                  <value><int>35</int></value>
               </data>
            </array>
         </value>

      </data>
   </array>
</value>
```

A simple struct might look like:

```
<value>
   <struct>
      <member>
         <name>givenName</name>
         <value><string>Joseph</string></value>
      </member>

      <member>
         <name>familyName</name>
         <value><string>DiNardo</string></value>
      </member>

      <member>
         <name>age</name>
         <value><int>27</int></value>
      </member>
   </struct>
</value>
```

This way you can implement almost all data types supported by any programming language.