



Distributed Programming - xmlrpc - the specification

- Marc Conrad
 - D104a (Park Square Building)
 - Marc.Conrad@luton.ac.uk

 - Resources:
 - www.xmlrpc.com
 - Blackboard
 - Today: The XML specification

Overview

- XML-RPC is a Remote Procedure Calling protocol that works over the Internet. An XML-RPC message is an HTTP-POST request. The body of the request is in XML. A procedure executes on the server and the value it returns is also formatted in XML.
- Procedure parameters can be scalars, numbers, strings, dates, etc.;

*But no Objects!**

*Objects as in "Object Oriented Programming"



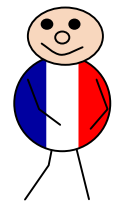
Request example

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: cis69.dyndns.com
Content-Type: text/xml
Content-length: 144
```

The Header

```
<?xml version="1.0"?>
<methodCall>
  <methodName>examples.name</methodName>
  <params>
    <param>
      <value><i4>41</i4></value>
    </param>
  </params>
</methodCall>
```

examples.name(41)



Header requirements

- The format of the URI in the first line of the header is not specified. For example, it could be empty, a single slash, if the server is only handling XML-RPC calls. However, if the server is handling a mix of incoming HTTP requests, we allow the URI to help route the request to the code that handles XML-RPC requests. (In the example, the URI is /RPC2, telling the server to route the request to the "RPC2" responder.)
- A User-Agent and Host must be specified.
- The Content-Type is text/xml.
- The Content-Length must be specified and must be correct.

Payload format

- The payload is in XML, a single `<methodCall>` structure.
- The `<methodCall>` must contain a `<methodName>` sub-item, a string, containing the name of the method to be called. The string may only contain identifier characters, upper and lower-case A-Z, the numeric characters, 0-9, underscore, dot, colon and slash. It's entirely up to the server to decide how to interpret the characters in a `methodName`.
- For example, the `methodName` could be the name of a file containing a script that executes on an incoming request. It could be the name of a cell in a database table. Or it could be a path to a file contained within a hierarchy of folders and files.
- If the procedure call has parameters, the `<methodCall>` must contain a `<params>` sub-item. The `<params>` sub-item can contain any number of `<param>`s, each of which has a `<value>`.



Scalar <value>s

- <value>s can be scalars, type is indicated by nesting the value inside one of the tags:
- <i4> or <int> (four-byte signed integer), e.g: 3, 888, -12, 0
- <boolean> , e.g. 0 (false) or 1 (true)
- <string> , e.g. hello world, Marc Conrad
- <double> (floating point number), e.g. -12.214
- <dateTime.iso8601> (date/time) 20031017T14:08:55
- <base64> (base64-encoded binary) eW91IGNhbidpcyE=
If no type is indicated, the type is string.

A value can also be of type <struct>.

- A <struct> contains <member>s and each <member> contains a <name> and a <value>.
- Example of a two-element <struct>:

```
<struct>
```

```
<member>
```

```
<name>lowerBound</name>
```

```
<value><i4>18</i4></value>
```

```
</member>
```

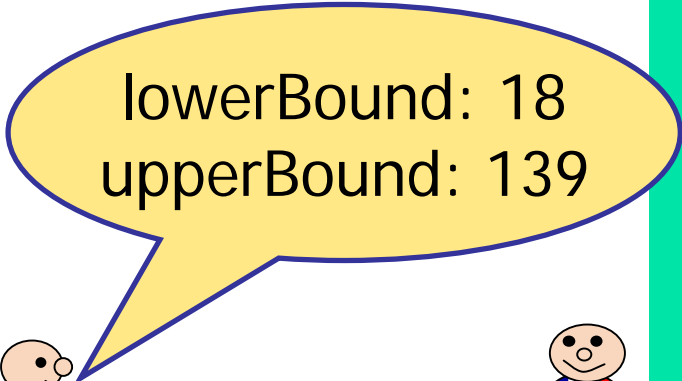
```
<member>
```

```
<name>upperBound</name>
```

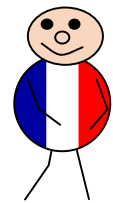
```
<value><i4>139</i4></value>
```

```
</member>
```

```
</struct>
```



lowerBound: 18
upperBound: 139



A value can also be of type <array>.

- An <array> contains a single <data> element, which can contain any number of <value>s.
- Here's an example of a four-element array:

```
<array>
```

```
<data>
```

```
<value><i4>12</i4></value>
```

```
<value><string>Egypt</string></value>
```

```
<value><boolean>0</boolean></value>
```

```
<value><i4>-31</i4></value>
```

```
</data>
```

```
</array>
```

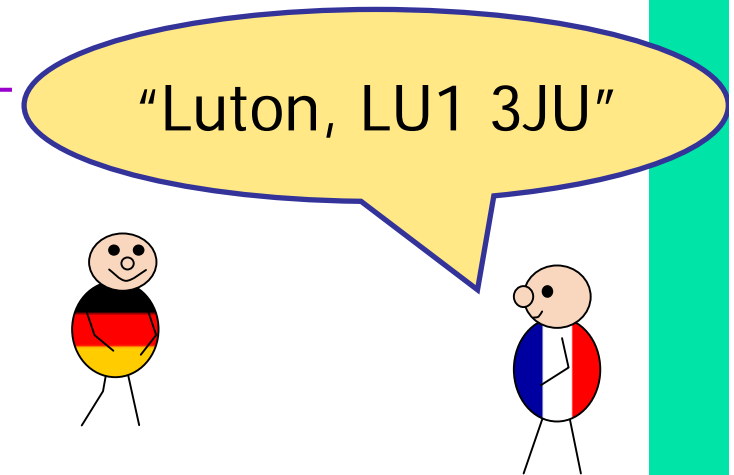
[12,"Egypt",false,-31]



Here's an example of a response to an XML-RPC request:

HTTP/1.1 200 OK
Connection: close
Content-Length: 134
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT

```
<?xml version="1.0"?>  
<methodResponse>  
  <params>  
    <param>  
      <value><string>Luton, LU1 3JU</string></value>  
    </param>  
  </params>  
</methodResponse>
```



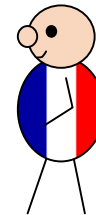
Response format

- Unless there's a lower-level error, always return 200 OK.
- The Content-Type is text/xml. Content-Length must be present and correct.
- The body of the response is a single XML structure, a `<methodResponse>`, which can contain a single `<params>` which contains a single `<param>` which contains a single `<value>`.
- The `<methodResponse>` could also contain a `<fault>` which contains a `<value>` which is a `<struct>` containing two elements, one named *faultCode*, an `<int>` and one named *faultString*, a `<string>`.

Fault example

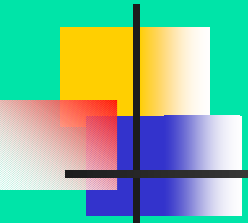
```
<?xml version="1.0"?> <methodResponse>  
  <fault>  
    <value>  
      <struct>  
        <member>  
          <name>faultCode</name>  
          <value><int>4</int></value>  
        </member>  
        <member>  
          <name>faultString</name>  
          <value><string>Overflow</string> </value>  
        </member>  
      </struct>  
    </value>  
  </fault>
```

faultCode: 4
faultString: Overflow



Copyright of the Specification

- © Copyright 1998-2003 UserLand Software. All Rights Reserved.
- This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and these paragraphs are included on all such copies and derivative works.



Copyright of the Specification (cont'd)

- This document may not be modified in any way, such as by removing the copyright notice or references to UserLand or other organizations. Further, while these copyright restrictions apply to the written XML-RPC specification, no claim of ownership is made by UserLand to the protocol it describes. Any party may, for commercial or non-commercial purposes, implement this protocol without royalty or license fee to UserLand. The limited permissions granted herein are perpetual and will not be revoked by UserLand or its successors or assigns.