

XML-RPC vs. SOAP

by kate rhodes

masukomi@masukomi.org

**A simple guide to choosing the best protocol
for your XML Remote Procedure Call
needs.**

Within the world of XML there are two main ways to implement a Remote Procedure Call (RPC). XML-RPC and SOAP. This document will explore the differences between these two methods in order to help you decide which is best suited to your needs.

General information

XML-RPC <http://www.xmlrpc.com>

XML-RPC is

“...a spec (<http://www.xmlrpc.com/spec>) and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet.

It's remote procedure calling using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.” - xmlrpc.com

XML-RPC's Goals

XML-RPC is very humble in its goals. It doesn't set out to be the solution to every problem. Instead it seeks to be a simple and effective means to request and receive information.

“We wanted a clean, extensible format that's very simple. It should be possible for an HTML coder to be able to look at a file containing an XML-RPC procedure call, understand what it's doing, and be able to modify it and have it work on the first or second try... We also wanted it to be an easy to implement protocol that could quickly be adapted to run in other environments or on other operating systems.” - xmlrpc.com

Features & Benefits

Comments

The spec itself is roughly seven pages long, including examples and a FAQ, and is extremely easy to understand. Any competent programmer should find no difficulty whatsoever in implementing XML-RPC in their software after reading its spec.

SOAP 1.1 <http://www.w3.org/TR/SOAP/>

“SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.” - the SOAP spec.

SOAP makes extensive use of namespacing and attribute specification tags in almost every element of a message. For example when mixing data types within an array you have to set the SOAP-ENC:arrayType to indicate mixed data types within the array in addition to specifying the type of each element of the array.

SOAP's Goals

SOAP tries to pick up where XML-RPC left off by implementing user defined data types, the ability to specify the recipient, message specific processing control, and other features.

Comments

Weighing in at 40 pages the SOAP spec is complex and filled with little gems like, “Using SOAP for RPC is orthogonal to the SOAP protocol binding.” If you ask me, it isn't even remotely something I would call “lightweight” and they threw out the most important feature of XML-RPC, its ease of use.

Features & Benefits

XML-RPC

Datatypes

Scalars:

Type	Example
32-bit signed integer	-12
boolean	0 or 1
ASCII string	hello world
double-precision signed floating point number	-12.214
date/time (iso 8601)	19980717T14:08:55
base64-encoded binary	eW91IGNhbid0IHJlYWQgdGhpcyE=

Structs:

Features & Benefits

In XML-RPC structs define an anonymous set of name value pairs. "A <struct> contains <member>s and each <member> contains a <name> and a <value>. "The value(s) can be of any data type

Arrays:

XML-RPC arrays define an anonymous grouping of elements with no limitation mixing data types like integers and strings within the same array. "An <array> contains a single <data> element, which can contain any number of <value>s."

Stability

XML-RPC has been around for a while and, while it isn't maintained by a standards committee, it is stable and open to community input.

Simplicity

XML-RPC's simplicity is its greatest feature. It is extremely easy to understand, implement, and debug. The syntax is so uncomplicated that it is very easy to find, and avoid, mistakes.

SOAP

Datatypes

Scalars:

Type	Example
32-bit signed integer	-12
boolean	0 or 1
ASCII string	hello world
signed floating point number	-12.214
date/time	2001-03-27T00:00:01-08:00
base64-encoded binary	eW91IGNhbid0IHJlYWQgdGhpcyE=

Structs:

SOAP structs define a set of name value pairs. Structs can be named.

Arrays:

SOAP arrays define a grouping of elements with no limitation mixing data types like integers and strings within the same array. Arrays can be named.

Array of Bytes:

An array of bytes MAY be encoded as a single-reference or a multi-reference value. The rules for an array of bytes are similar to those for a string.

In particular, the containing element of the array of bytes value MAY have an "id" attribute. Additional accessor elements MAY then have matching "href" attributes."

Limitations

Polymorphic Accessors:

An accessor "...that can polymorphically access values of several types, each type being available at run time. A polymorphic accessor instance MUST contain an "xsi:type" attribute that describes the type of the actual value."

Ex. `<cost xsi:type="xsd:float">29.95</cost>`

Enumerations:

Based on the "XML Schema Part 2: Datatypes" (<http://www.w3.org/TR/xmlschema-2/>), which is still just a recommendation, a SOAP enumeration is "...a list of distinct values appropriate to the base type." It is supported for all simple types except boolean.

User Defined Data-Types:

Developers can define their own simple, or complex, data types.

Customizing

SOAP's greatest feature is its ability to step past XML-RPC's limitations and customize every portion of the message. This ability to customize allows developers to describe exactly what they want within their message. The downside of this is that the more you customize a message the more work it will take to make a foreign system do anything beyond simply parsing it.

Big Co. Support

SOAPs two biggest developers are Microsoft and IBM. Microsoft has incorporated SOAP into it's latest OS and Visual Studio. As usual though, MS has decided that they don't need to adhere to the whole spec. In their SOAP for Java SDK they modify the namespace (which breaks some other implementations), use a limited combination of 1.0 and 1.1 fault codes (and improperly document it), and include many serious bugs. IBM seems to have gotten almost everything right in their implementation.

Limitations

XML-RPC

Method Calls

XML-RPC calls methods via its methodName property which "may only contain identifier characters, upper and lower-case A-Z, the numeric characters, 0-9, underscore, dot, colon and slash." For most purposes this is just fine, however it makes it particularly difficult when you need to pass an object as an argument.

Named Data Structures

The structs (hashes) and arrays are always anonymous. When passing multiple structs or arrays programmers need to rely on the order of the parameters (<param>) they are contained in to differentiate between each struct or array. This isn't a significant problem but there are cases when being able to name your structs and arrays would be nice.

Simplicity

As noted above, simplicity is also XML-RPC's greatest limitation. Although most all of your RPC needs can be accomplished with it, there are some things that you just can't

Comparison

do without bending over backwards, like passing an object as an argument to a function, or specifying which portion of a receiving application the message is intended for.

SOAP

Stability

Currently SOAP 1.1 is a *submission* to the W3C (<http://www.w3.org>). This means it is *not* an official standard and its specifications can, and almost assuredly will, change at any time. There are a lot of cooks in the kitchen right now.

Documentation

Last updated in May the SOAP spec is filled with mistakes and contradictions and points to other specs which are themselves moving targets. According to Dave Winer, one of the spec's authors, Microsoft and IBM have taken it upon themselves to add in the WSDL (<http://www.w3.org/TR/wsdl>) layer without consulting the other authors. This doesn't bode well for developers who need something they can count on remaining unchanged, or almost unchanged, for the foreseeable future. Any current SOAP implementation may be incompatible with future revisions of the spec, although it should be noted that SOAP is far enough along that most of its spec should remain essentially the same.

Comparison

Feature	XML-RPC	SOAP
basic scalars	yes	yes
structs	yes	yes
arrays	yes	yes
named structs and arrays	no	yes
detailed fault handling	yes	yes
short learning curve	yes	no
Developers specified character set	no	yes (US-ASCII, UTF-8, UTF-16)
Developer defined data types	no	yes
Can specify recipient	no	yes
require client understanding	no	yes
message specific processing instructions	no	yes

When you get right down to it XML-RPC is about simple, easy to understand, requests and responses. It is a lowest common denominator form of communication that allows you to get almost any job done with a minimum amount of complexity. SOAP, on the

Sources & Other Links

other hand, is designed for transferring far more complex sets of information. It requires profuse attribute specification tags, namespaces, and other complexities, to describe exactly what is being sent. This has its advantages and disadvantages. SOAP involves significantly more overhead but adds much more information about what is being sent. If you require complex user defined data types and the ability to have each message define how it should be processed then SOAP is a better solution than XML-RPC (be sure to check out language specific solutions to this problem like java's RMI (<http://java.sun.com/products/jdk/rmi/>)). But, if standard data types and simple method calls are all you need then XML-RPC will give you a faster app with far fewer headaches.

“Make sure you don't use an overly complex package to address what may be your rather simple needs” -Brett McLaughlin (brett@newinstance.com)

Sources & Other Links

XML-RPC

- XML-RPC Specification
<http://www.xmlrpc.com/spec>
- Internet Scripting: Zope and XML-RPC
<http://www.xml.com/pub/a/2000/01/xmlrpc/>

SOAP

- SOAP specification
<http://www.w3.org/TR/SOAP/>
- SoapWare.org A busy developers guide to SOAP 1.1
<http://www.soapware.org/bdg>
- Some considerations on SOAP
[http://www.xmlrpc.com/discuss/msgReader\\$1500?mode=day](http://www.xmlrpc.com/discuss/msgReader$1500?mode=day)

Assorted other links

- PI-RPC, the Platform Independent Remote Procedure Call
<http://www.blackperl.com/xml/PI-RPC.html>
is a proposed lightweight alternative to SOAP.
- XML-RPC Extensions
<http://ontosys.com/xml-rpc/extensions.html>
add a <nil/> value.
- The Evolution of SOAP::Lite
http://www.onlamp.com/pub/a/onlamp/2001/06/29/soap_lite.html
- WSDL 1.1
<http://www.w3.org/TR/wsdl>