SOAP –STANDARD FOR WEB SERVICIES

Assoc.Prof. Ph.D. Eng. ADRIANA BARNOSCHI
University "N. Titulescu" Romania
abarnoschi2002@yahoo.com

Abstract
SOAP, the Simple Object Access Protocol, is the newest toy to come from the World Wide Web Consortium and Microsoft. The communication is made up of a SOAP client and a SOAP server.
SOAP implementations can be optimized to exploit the unique characteristics of particular network systems. All SOAP messages are encoded using XML. Web Services continue the evolution toward lower cost, easier to use standard-based B2Bi tools and interfaces.

1. Introduction

Short for ***S****imple* ***O****bject* ***A****ccess* ***P****rotocol*, a lightweight XML-based messaging protocol used to encode the information in Web service request and response messages before sending them over a network. SOAP messages are independent of any operating system or protocol and may be transported using a variety of Internet protocols, including SMTP, MIME, and HTTP. Based on XML, it allows the transfer of data through the HTTP headers. This means you can conduct all you transfer on port 80, therefore preventing the opening of any security holes on your server.

SOAP consists of two files: Client and Server (the basic model in Internet), and its dependant on the XMLDOM and XMLHTTP objects.

SOAP is a standard means of sending communications from one Web server to another, regardless of platform or Web server software being used. SOAP messages a standard designed to enable server communications via an XML-formatted message through the HTTP protocol. The SOAP client sends a SOAP request to the SOAP server, which, in turn, sends back a SOAP response.

SOAP consists of three parts:

1. The SOAP *envelope* - construct defines an overall framework for expressing what is in a message; who should deal with it, and whether it is optional or mandatory.
2. The SOAP *encoding rules* - defines a serialization mechanism that can be used to exchange instances of application-defined datatypes.
3. The SOAP *RPC representation* - defines a convention that can be used to represent remote procedure calls and responses.

Although these parts are described together as part of SOAP, they are functionally orthogonal. In particular, the envelope and the encoding rules are defined in different namespaces in order to promote simplicity through modularity.

In addition to the SOAP envelope, the SOAP encoding rules and the SOAP RPC conventions, this specification defines two protocol bindings that describe how a SOAP message can be carried in HTTP messages either with or without the HTTP Extension Framework.

A major design goal for SOAP is simplicity and extensibility. This means that there are several features from traditional messaging systems and distributed object systems that are not part of the core SOAP specification. Such features include

- ✓ Distributed garbage collection
- ✓ Boxcarring or batching of messages

- ✓ Objects-by-reference (which requires distributed garbage collection)
- ✓ Activation (which requires objects-by-reference)

2. The SOAP Message Exchange Model

SOAP messages are fundamentally one-way transmissions from a sender to a receiver, but as illustrated above, SOAP messages are often combined to implement patterns such as request/response.

Regardless of the protocol to which SOAP is bound, messages are routed along a so-called "message path", which allows for processing at one or more intermediate nodes in addition to the ultimate destination.

A SOAP application receiving a SOAP message MUST process that message by performing the following actions in the order listed below:

1. Identify all parts of the SOAP message intended for that application
2. Verify that all mandatory parts identified in step 1 are supported by the application for this message and process them accordingly. If this is not the case then discard the message. The processor MAY ignore optional parts identified in step 1 without affecting the outcome of the processing.
3. If the SOAP application is not the ultimate destination of the message then remove all parts identified in step 1 before forwarding the message.

Processing a message or a part of a message requires that the SOAP processor understands, among other things, the exchange pattern being used (one way, request/response, multicast, etc.), the role of the recipient in that pattern, the employment (if any) of RPC mechanisms, the representation or encoding of data, as well as other semantics necessary for correct processing.

While attributes such as the SOAP encodingStyle attribute can be used to describe certain aspects of a message, this specification does not mandate a particular means by which the recipient makes such determinations in general.

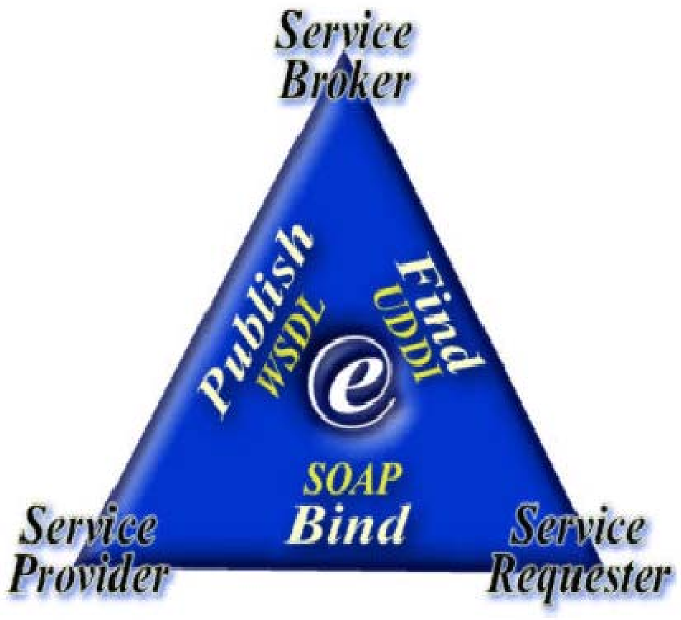
SOAP naturally follows the HTTP request/response message model providing SOAP request parameters in a HTTP request and SOAP response parameters in a HTTP response. Note, however, that SOAP intermediaries are NOT the same as HTTP intermediaries. That is, an HTTP intermediary addressed with the HTTP Connection header field cannot be expected to inspect or process the SOAP entity body carried in the HTTP request.

3. The Web Service Model

Simple: XML, HTTP
Dynamic: Publish, Find, Bind
Standard: XML, XML Schema, SOAP, WSDL, UDDI, ...

- ➢ **Standards**

TCP/IP, HTTP, XML, ...



- **Industry Support**

The Internet, routers, load balancing, firewalls, web servers, ...

The Web Services Stack

- HTTP
- XML
    - ✓ XML Schema
    - ✓ SOAP - Simple Object Access Protocol
    - ✓ WSDL - Web Services Description Language
    - ✓ UDDI - Universal Description, Discovery and Integration
- Java

A **web service** consists of three components:

1. a listener to receive the message,
2. a proxy to take that message and translate it into an action to be carried out (such as invoking a method on a Java object), and
3. the application code to implement that action.

The listener and proxy components should be completely transparent to the application code, if properly implemented. The ideal situation in most cases is that the code doesn't even know it is being invoked through a web service interface, but that is not always possible, or desirable.

There is an unexpectedly long list of SOAP implementations available to developers. The most popular tools: Apache SOAP for Java, SOAP::Lite for Perl, and Microsoft .NET. No matter which toolkit you use, the fundamental process of creating, deploying, and using SOAP web services is the same. A good example of a seamless, simple web services implementation is the SOAP::Lite for Perl written by Pavel Kulchenko.

4. Handling SOAP Messages

The integration of SOAP toolkits varies with the transport layer. Some implement their own HTTP servers. Some expect to be installed as part of a particular web server, so that rather than serving up a web page, the HTTP daemon hands the SOAP message to the toolkit's proxy component, which does the work of invoking the code behind the web service (Figure 1).

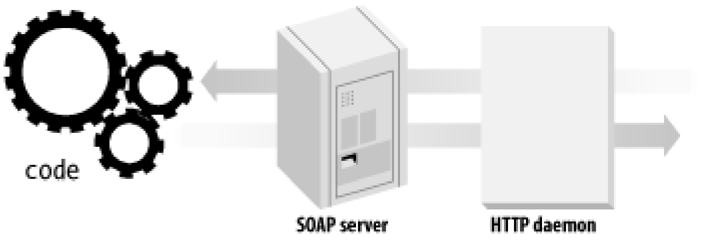


Figure 1. The HTTP daemon passes the request to the SOAP proxy, which then invokes the code behind the web service

Proxy component (when it handed a SOAP message by a listener) must do following:

1. **Deserialize the message,** if necessary, from XML into some native format suitable for passing off to the code.

2. **Invoke the code.** Serialize the response to the message (if one exists) back into XML and hand it back to the transport listener for delivery back to the requester.
3. **Deploying Web Services.** Deploying a web service involves telling the proxy component which code to invoke when a particular type of message is received. In other words, the proxy component has to know that a getQuote message is going to be handled by the samples. QuoteServer Java class or the QuoteServer.pm Perl module. Once this has happened, clients can access the server, send the message, and trigger a call to application code.

Creating web services in Java is more work than in Perl with SOAP::Lite, but the process is essentially the same. Apache SOAP is the Apache Software Foundation's implementation of the SOAP protocol. It is designed to run as a servlet within any Java HTTP Server. As such, it implements only the proxy part of the message handling process. Like SOAP::Lite, Apache SOAP's list of features is impressive, sharing many of the same benefits as its Perl-based counterpart.

More Web Services Support
WebSphere Business Components
- ➢ Enabled for UDDI, SOAP and other Web Services standards in 2001

MQ Series Family
- ➢ SOAP support in 1H2001

Lotus
- ➢ Direction to enable Domino services as Web services, by incorporating

SOAP interfaces and XML-based messaging

5. Conclusions

Developers who are writing and using web services that may be accessed by a wide variety of SOAP implementations need to be aware that inconsistencies like this will exist between the various toolkits and you need to be prepared to deal with them. As web services become more complex and more missions critical, it is important to have a clear understanding of how to manage these issues.

A SOAP application SHOULD include the proper SOAP namespace on all elements and attributes defined by SOAP in messages that it generates. A SOAP application MUST be able to process SOAP namespaces in messages that it receives. It MUST discard messages that have incorrect namespaces and it MAY process SOAP messages without SOAP namespaces as though they had the correct SOAP namespaces.

6. References

1. James Snell, Doug Tidwell & Pavel Kulchenko, “Programming Web Services with SOAP”, 1st Edition December 2001, cap.3
2. www.webopedia.com/term
3. Erick Stover, “SOAP: Platform-Independent Server Communication”
4. W3C SOAP Protocol
5. Bradner S., "The Internet Standards Process -- Revision 3", RFC2026, Harvard University, October 1996
6. Barnoschi A., “Web Servicies – processes of the integration in affairs”, The 31st Internatonally Attented Scientific Conference, Military Technical Academy, Bucharest, 2005, pg.125-129