Simple Object Access Protocol
A basic overview
Distributed Systems Fall 2002

Tom Wennerström
towe5275@student.uu.se

Jens Jespersen
jeje5651@student.uu.se

Magnus Lundquist
malu0843@student.uu.se

September 30, 2002
Abstract

SOAP or Simple Object Access Protocol, is a lightweight XML based protocol for exchanging information in a decentralized distributed environment. It supports a Remote Procedure Call style and also message oriented data exchange. SOAP has derived from the XML-RPC standard, and all SOAP messages are encoded with XML. A SOAP message is typically transported via HTTP. This makes it firewall-friendly and suitable for usage over the internet. A SOAP message consists of three basic blocks. An envelope, a header and a body. Because of the usage of HTTP, any security mechanisms that can be used on a regular HTTP message can also be applied on a SOAP message. The problem is to separate a SOAP message from any other HTTP messages in e.g. a firewall. This mainly puts the security aspects in the hands of the application developer. And if we look at the efficiency we quickly discover that SOAP may not be the most efficient protocol.

Keywords: soap, web service, xml, security, rpc
## Contents

1 What is SOAP? 2
  1.1 SOAP, what’s behind it? 2
  1.2 A simple SOAP message 2
  1.3 The SOAP message exchange model 4
    1.3.1 RPC 4
    1.3.2 Message Chains 5
  1.4 SOAP usage in Web Services 5

2 What are the limitations of SOAP? 6

3 SOAP and Security 6

4 Is SOAP the answer to everything? 7

5 Conclusion 7
1 What is SOAP?

The Simple Object Access Protocol, or short SOAP, is a lightweight XML based protocol for exchanging information in a decentralized distributed environment. [3] It is developed by Microsoft Development, Userland Software and IBM. The specification of SOAP has been established by the W3C organization. W3C was founded in 1994 and is supported by several companies worldwide.

SOAP supports different styles of information exchanged. It supports a Remote Procedure Call (RPC) style, which allows a request-response processing. And it also supports message oriented information exchange. This is good if you like to exchange business documents or other types of documents where the sender may not expect to get a response immediately.

SOAP also have the following features (to mention some):

- Protocol independence
- Language independence
- Platform and operating system independence
- Simple and extensible

1.1 SOAP, what’s behind it?

The key making all this possible is XML. XML, or eXtensible Markup Language, is [10] a markup-language for documents containing structured information. XML is, in contrast to HTML, not designed to display data. Instead the main purpose of XML is to describe data. Like HTML, XML uses the HTTP protocol as the transport protocol. SOAP has derived from an earlier XML based standard, the XML-RPC, and all SOAP messages are encoded using XML. Because of the fact that SOAP uses the HTTP protocol (mostly) it is very firewall friendly. This is a desirable thing. It makes SOAP easy to use over the internet.

| Simple Object Access Protocol (SOAP) |
| Extensible Markup Language (XML) |
| Application Layer (HTTP, FTP, ...) |
| Transport Layer (TCP, UDP, ...) |
| Network Layer (IP) |
| Link Layer (Ethernet, ...) |
| Physical Layer (Hardware) |

Table 1: SOAP's location in the protocol-stack

Table 1 shows the location of the SOAP in the typical protocol-stack

1.2 A simple SOAP message

A SOAP message consists of three different parts:

- An Envelope that defines the general structure and information of the message. It tells you who is going to handle the message, and if it is voluntary or compulsory.

- A Header, which may be present in a SOAP message. The header consists of rules of encoding, definitions of datatypes to mention a few things.
• A Body, which must be present in a SOAP message. The body typically consists of marshalled RPC calls and error reporting.

![Diagram of SOAP message]

Figure 1: A SOAP message

To get a feeling for how a SOAP message can look like here is a quick example. This example is also meant to point out that a SOAP message is almost in plain English. In this example, a GetLastTradePrice SOAP request is sent to a StockQuote service. The request takes a string parameter, ticker symbol, and returns a float in the SOAP response. [1]

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/”>
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

And the following is a fictive response to this message containing the HTTP message with the SOAP message as the payload. Notice that in this example there is no header, and that the message consists only of an envelop and a body.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
```
Exactly how everything in this example works and why things look like they do are not going to be covered in this paper, as it has no value to the general understanding of SOAP.

1.3 The SOAP message exchange model

SOAP is fundamentally a one-way transmission protocol [1], from a sender to a receiver. But often it is used as described previously, as a reply-response pattern.

1.3.1 RPC

For an example let’s look at a simple database:

In the picture above, application B represents the database and application A the client that’s making a request. The SOAP approach to this problem is to [7]

- Create a interface for the request-logic of a method in the database. This method could be written in any language, e.g. C/C++.

- Set up a listening-process that listens to incoming requests with the SOAP inteface.

This listening-process decodes the incoming SOAP request and transforms it into an invocation of the method. It then takes the result of the method called, encodes it into a SOAP message and sends it back to the client who invoked the method.
1.3.2 Message Chains

Because SOAP is initially based on a one-way communication, other useful tools than RPC can be used. One example of this is Message Chains. The main thing in this approach are logical units called Endpoints. Endpoints are receivers of SOAP messages. It is the responsibility of an endpoint to examine a message and remove the part that was addressed to that endpoint for processing.

Endpoints can function as both senders and receivers. These are called Intermediaries and can be used as routers. They route messages in the chain with some potential processing at each step. With this technique, scenarios like those in Fig.3 can be achieved.

```
+-----------------+            +-----------------+
|  Request        |            |  Response       |
|                 +-----------------+            +-----------------+
|                 |            | Broadcast       |
|                 +-----------------+            +-----------------+
|  Sender         |            |  EndPoint A     |
|                 +-----------------+            +-----------------+
|                 |            |  EndPoint B     |
|                 +-----------------+            +-----------------+
|                 |            |  EndPoint C     |
|                 +-----------------+            +-----------------+
|  EndPoint C     |            |  EndPoint A     |
|                 +-----------------+            +-----------------+
|  EndPoint B     |            |  EndPoint A     |
```

Figure 3: Examples of message chains

1.4 SOAP usage in Web Services

The Service Web was created when the e-businesses wanted to integrate their processes with other companies. [8]

The Service Web is powered by Web application servers that speaks SOAP, and deliver information marked up in XML. The building block of the Service Web is the Web Service, a set of related application functions that can be invoked over the internet. To invoke a Web Service the application needs information about the service, which is given by a Web Service Description Language (WSDL) document. The WSDL documents are indexed in searchable Universal Description Discovery, and Integration business registers (UDDI) that tell were the Web Services are located. Web services is a way to connect different systems and programs with each other over the Internet, despite differences in programming language or system implementation. This will make it much easier and cheaper for companies to interact with each other and to offer different services.

- Application server hosts Web Services and makes them accessible using protocols such as HTTP GET, HTTP POST and SOAP/HTTP.
- XML Repository contains WSDL documents that describes the Web Services.
- Business registry contains the indexed UDDI, which contains the URL:s of the WSDL documents.
2 What are the limitations of SOAP?

SOAP does not say anything about bi-directional communication, although it is possible to layer these semantics on top of a SOAP implementation. [4] The current SOAP specification describes how a SOAP payload can be transmitted via HTTP, but does not address any other protocols. SOAP does not address higher-level issues such as object activation or lifetime management. SOAP does not mandate any particular language for interface description, although the XML specification is a reasonable way to describe interfaces and user-defined types.

3 SOAP and Security

Because SOAP messages mainly is passed through HTTP all security mechanism used in HTTP can be used for SOAP messages. This means that you can use e.g. SSL or HTTPS with SOAP without problems.

Because SOAP is simply another payload carried via HTTP any potential vulnerabilities are similar as with straight HTTP. This may be right, but the problem is that if a system administrator by some reason wants to configure a firewall to stop some SOAP messages from getting pass the firewall, he got problems. The only way to recognize a SOAP message is by doing a XML parsing.

Okay, so now you know it is a SOAP message. And now you want to decide which SOAP messages to let pass through the firewall. SOAP has no reliable internal structure. Sometimes there is a header, sometimes there is not. Sometimes the body is RPC-encoded and sometimes it’s not. Sometimes there is a method name, and sometimes there is not. This may be a security risk, becase you can not, in an easy/efficient way, filter SOAP messages.

This puts all the security aspects in the hands of the application developers. And this may not be the perfect idea. Most application developers is not that intrested in the security aspect. [6]
4 Is SOAP the answer to everything?

According to McLaughlin [5] there isn’t always the need to use SOAP if you want to have simple RPC. Infact, if this is the only feature you need one can use XML-RPC with the same result, and more simplicity. The specification for SOAP does not say anything about RPC. This is gained through the underlying XML-RPC.

If we look at the efficiency, there is a major drawback in the message overhead. Because a SOAP message is sent in a human readable fashion, there will be a lot of redundancy in the message. This is generally not the case with protocols that send data in a binary non human readable fashion. This means that the communication over the network is going to suffer in loss of capacity.

One other thing that one might think of before settle for the SOAP as the RPC for your system is that if you e.g. have a distributed database and the data in it isn’t stored in XML you will have to spend a lot of time converting your data to XML for every message sent with SOAP. If this is the case, maybe you should consider switching either database storing method to XML, not use SOAP or if money is not an issue, buy really powerful machines.

So, what does SOAP provide that XML-RPC doesn’t? That is the following.

- An envelope which carries information about the included message.
- A set of rules for encoding application-specific data types.

This may be good, but as said before, in most cases the simpler XML-RPC is sufficient for most applications.

5 Conclusion

So, are there any conclusions here? Well, first of all there is one thing: SOAP is an addition to the XML-RPC. SOAP just adds some easy ways of handle strange or user-defined datatypes, and an envelope to the message. SOAP seems like a good idea, but looking closer there are some problems lurking in the horizon.

One of them is the security issue. The fact that SOAP is firewall friendly and usually is transported by HTTP makes thing difficult for administrators, if they want to monitor the SOAP messages to allow passage through the security. The other one is that SOAP is in most cases an overkill. That is in message overhead and functionality. For most cases the simpler XML-RPC is sufficient. And the last obvious drawback is all encoding/decoding from XML to application-data (if the application don’t use XML as datastructure from the beginning). This mean a raise of computation for the system.

But there are some good things about it. First of all, SOAP is, in theory, platform independent. There are programming tools for almost any language and platform. The second thing is the simplicity of usage. It is a human-readable protocol and it is easy to work with.
References


