SOAP : Simple Object Access Protocol

- SOAP is an XML-based messaging protocol specification for exchanging structured information in the implementation of web services.
- SOAP enables web applications to exchange messages regardless of the languages, system platforms, operating systems used for application development.

SOAP

- SOAP is more reliable than the convectional distributed communication technologies (CORBA, RMI, DCOM)
- Defines a way of passing XML-encoded data, including binary content.
- Weaknesses of ORPC/ Convention distributed communication

– Both ends of the communication link would need to be implemented under the same distributed object model (Java/RMI or CORBA/IIOP)

– Difficulty of getting these protocols to work over firewalls or proxy servers, e.g, most firewalls are configured to allow hypertext transfer protocol (HTTP) to pass across, but not IIOP.

SOAP

- To address the problem of overcoming proprietary systems running on heterogeneous infrastructures, Web services rely on SOAP, an XML based communication protocol for exchanging messages between computers regardless of their operating systems, programming environment or object model framework.

Where did SOAP come from?

- *Coauthored by representatives of IBM, Lotus, Microsoft, DevelopMentor, and UserLand.*
- Later submitted to the World Wide Web Consortium (W3C).
- Current SOAP development is the domain of the W3C XML Protocol Working Group

How does SOAP work?

- Implements the request/response model of HTTP
- A client sends a SOAP message to a server that processes the **request** and **responds** with a SOAP message of its own.
- SOAP requests and responses *travel* using HTTP, HTTPS, or some other transport mechanism.

HOW does SOAP work?

- In general, a SOAP service remote procedure call (RPC) request/response sequence includes the following steps:
- A ***SOAP client*** formulates a **request for a service**. This involves creating a conforming XML document, either explicitly or using specific SOAP client API.
- A SOAP client sends the **XML document** to a *SOAP server*. This SOAP request is posted using HTTP or HTTPS to a SOAP Request Handler running as a servlet on a Web server.

How does SOAP work?

Example 1-1 shows the body of a SOAP message, an XML document, that represents a SOAP request for a service that provides an address from an address book.

- The Web server receives the SOAP message, an XML document, using the SOAP Request Handler Servlet. The server then dispatches the message as a service invocation to an appropriate server-side application providing the requested service.
- A response from the service is returned to the SOAP Request Handler Servlet and then to the caller using the standard SOAP XML payload format. Example 1-2 contains the body of a response to the request made in Example 1-1

How does SOAP work?

Example 1-1 SOAP Request for Address Book Listing Service

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:getAddressFromName xmlns:ns1="urn:www-oracle-com:AddressBook"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<nameToLookup xsi:type="xsd:string" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
John B. Good
</nameToLookup>
</ns1:getAddressFromName>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

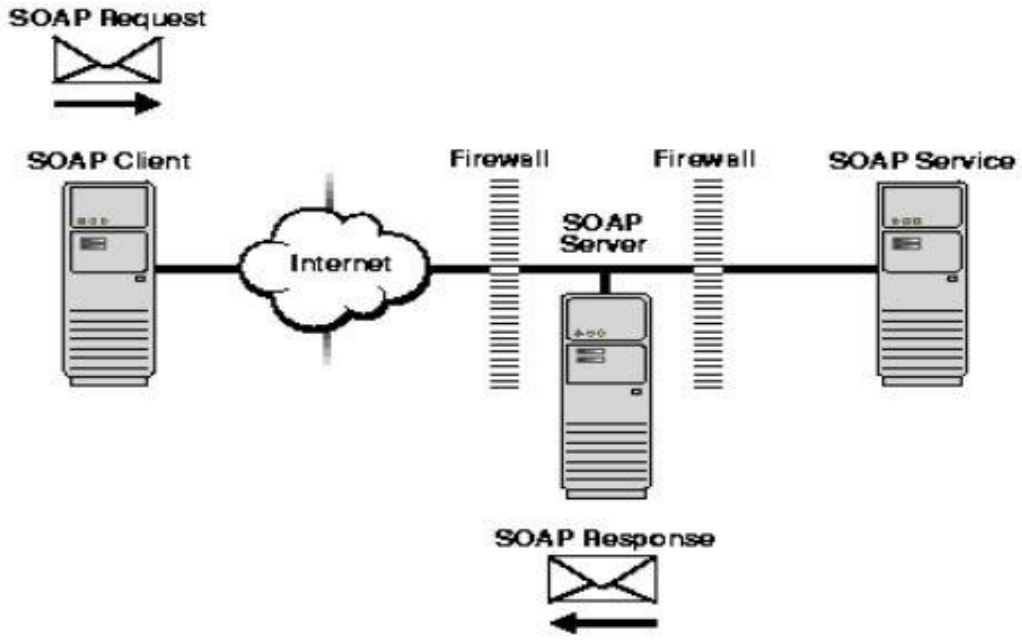
How does SOAP work?

Example 1-2 SOAP Response from Address Book Service

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:getAddressFromNameResponse xmlns:ns1="urn:www-oracle-com:AddressBook"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<return xmlns:ns2="urn:xml-soap-address-demo" xsi:type="ns2:address">
<city xsi:type="xsd:string">Anytown
</city>
<state xsi:type="xsd:string">NY
</state>
<phoneNumber xsi:type="ns2:phone">
<areaCode xsi:type="xsd:int">123
</areaCode>
<number xsi:type="xsd:string">7890
</number>
<exchange xsi:type="xsd:string">456
</exchange>
</phoneNumber>
<streetName xsi:type="xsd:string">Main Street
</streetName>
<zip xsi:type="xsd:int">12345</zip>
<streetNum xsi:type="xsd:int">123
</streetNum>
</return>
</ns1:getAddressFromNameResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Architecture

Figure 1-1 Components of the SOAP Architecture



What is a SOAP Message?

- SOAP is based on **message exchanges.**
- Messages are seen as **envelopes** where the application encloses the data to be sent.
- A SOAP message consists of an **<Envelope>** element containing an optional **<Header>** and a mandatory**<Body>** element.

N.B The contents of these elements are application defined and not a part of the SOAP specification.

SOAP Header

- A SOAP <Header> contains blocks of information relevant to how the message is to be processed.

This helps pass information in SOAP messages that is not for the application but for the SOAP engine.

SOAP Envelope and Header

```
<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        ……
</env:Envelope>
```

Example of SOAP envelope

```
<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
      …
    <env:Header>
       <tx:transaction-id
           xmlns:tx="http://www.transaction.com/transaction"
                       env:mustUnderstand="true">
                 512
            </tx:transaction-id>
       <notary:token xmlns:notary="http://www.notarization-services.com/token"
              env:mustUnderstand="true">
                GRAAL-5YF3
       </notary:token>
    </env:Header>
            ……
</env:Envelope>
```

Example of SOAP header

SOAP Body

- The SOAP body is the area of the SOAP message, where the application specific XML data (payload) being exchanged in the message is placed.
- The **<Body> element** contains either of the following:

– **Application-specific data:** is the information that is exchanged with a Web service. The SOAP <Body> is where the method call information and its related arguments are encoded. It is where the response to a method call is placed, and where error information can be stored.

– **fault message:** is used only when an error occurs.

Sample SOAP Message with Body

```
POST /weather HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-length: XXX
SOAPAction: "http://bouy.atlanticocean.com/weather"

<?xml version="1.0"?>
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
  <SOAP-ENV:Body>
    <weather:getWeatherConditions
          xmlns:weather="http://buoy.atlanticocean.com/">
      <latitude xsi:type="xsd:int">30</latitude>
      <longitude xsi:type="xsd:int">50</longitude>
    </weather:getWeatherConditions>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

What is all this junk?

```
…
<weather:getWeatherConditions
      xmlns:weather="http://buoy.atlanticocean.com/">
   <latitude xsi:type="xsd:int">30</latitude>
   <longitude xsi:type="xsd:int">50</latitude>
</weather:getWeatherConditions>
…
```

- The server has an **object** defined as ‘weather’.
- ‘weather’ has a **method** called

getWeatherConditions(int latitude, int longitude)

If we were processing data from the server and we wanted the weather conditions, we would request it by saying:

```
weather.getWeatherConditions( 30, 50 );
```

What could the response be?

```
<?xml version='1.0' ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <weather:conditions id="weather_id"
          xmlns:weather="http://bouy.atlanticocean.com/weather">
      <latitude>30 degrees</latitude>
      <longitude>50 degrees</longitude>
      <timestamp>04/12/2005 12:34:05</timestamp>
      <temperature>36 degrees</temperature>
      <description>It's looking like the perfect storm.</description>
      <!-- ... more conditions go here... -->
    </weather:conditions>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

How can SOAP send attachments?

```
...
<weather:conditions id="weather_id"
        xmlns:weather="http://bouy.atlanticocean.com/weather">
    ...
    <aerialSnapshot
        href="cid:30deg50deg.jpg@atlanticocean.com"/>
    <satelliteFeed
        href="cid:30deg50deg.mov@atlanticocean.com"/>
    <publication
        href="cid:AtlanticOcean.pdf@atlanticocean.com"/>

    ...
</weather:conditions>
  ...
```

http://www.whoi.edu/sbl/liteSite.do?litesiteid=2332&articleId=3898

Why use SOAP?

- Platform & Language independent.
- It is a W3C standard, a standard amongst the web services community, and it's "eXstensible"
- Allows for binary data attachments.
- Enforce processing requirements with the 'mustUnderstand' attribute.

Are there SOAP alternatives?

- XML-RPC
  - a precursor to SOAP
  - designed by Dave Winer of UserLand Software (also participated in the development of SOAP)
  - more lightweight and easier to implement
  - less full-featured
  - lacks support

Resources

Gentle Introduction to SOAP

A Busy Developer's Guide to SOAP 1.1

W3C Links:

SOAP Specification(W3C)

SOAP 1.2 Attachment(W3C)