# TOPIC NAME: SOAP

BY

Dasari Jayanth

2019BCS-016

## Overview of SOAP

SOAP stands for Simple Object Access Protocol, is an XML-based messaging protocol for information exchanging among computers, allowing distributed elements of an application to communicate. To understand the SOAP, one must understand the client-server environment and have some XML knowledge. It is an open standard.

We can carry out SOAP over a variety of lower-level protocols. SOAP supports addressing, security, and format independence through the concept of routing a message through nodes that perform different functions.

SOAP, using XML, provides a lightweight mechanism for structured and typed information exchange between peers in a decentralized environment. Instead of application semantics like a programming model, it offers modular packaging modules and encoding mechanisms for encoding data within modules, allowing SOAP to be used in a large variety of systems ranging from messaging systems to RPC.

## Why SOAP?

Microsoft created SOAP to provide a way to communicate between the web applications over the Internet, mainly via HTTP. The remote procedure calls transported via HTTP are the initial focus of SOAP. HTTP because all Internet browsers and servers support it. It is easy to understand SOAP as XML is a human-readable language, and SOAP is XML based.

The difference between other frameworks like CORBA, DCOM, and SOAP is that SOAP messages are written entirely in XML. SOAP is platform independent(can be any operating system) and language independent(can be any programming language). Its difficulty depends on the programming language being used.

## Features of SOAP

- SOAP is an application communication protocol, a format for sending and receiving messages, and can be used for broadcasting a message.
- For XML messaging, SOAP can extend HTTP.
- For web services, it provides data transport.
- Built-in Error handling.
- SOAP can call a remote procedure or directly exchange complete documents.
- SOAP is the XML way of defining what information is sent and how.
- SOAP enables client applications to connect to remote services and invoke remote methods easily.

SOAP consists of three parts:
1. The SOAP envelope construct defines an overall framework about what is in a message, who should deal with it, and whether it is optional or mandatory.
2. The SOAP encoding rules define a serialization mechanism that can be used to exchange instances of application-defined data types.
3. The SOAP RPC representation defines a convention that can be used to represent remote procedure calls and responses.

## SOAP Building Blocks

Whenever a method is called in the web service by a client application, the web service will automatically generate a SOAP message with the necessary details of the data sent from the web service to the client application.

A SOAP message is an ordinary XML document that contains the following:

### Envelope Element:

Envelope element is used to encapsulate the entire SOAP message. It identifies the XML document as a SOAP message. Every envelope element must have at least one soap body element, which is mandatory.

### Header Element:

Header element contains header information, any optional attributes of the message used in processing the message, either at an intermediary point or at the ultimate end-point. It can be used to contain information such as authentication information or the definition of complex data types. It is an optional element.

### Body Element:

Body element contains the XML data comprising the message being sent, call and response information. It is the main element that contains the definitions of web methods along with any parameter information if required. It is a mandatory element.

### Fault Element:

It is an optional element. It contains information about errors occurring and status while processing the message

## Basic Structure of SOAP Message

```
<?xml version="....."?>
<soap:Envelope>
<soap:Header>
...
</soap:Header>
<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>
</soap:Envelope>
```

All the elements declared above are in the default namespace for the SOAP envelope, and for the default namespace for SOAP encoding and data types, you can refer: http://www.w3.org/2003/05/soap-encoding.

## Syntax Rules

Here are some essential syntax rules:

- A SOAP message MUST be encoded using XML.
- A SOAP message MUST use the SOAP Envelope namespace.
- A SOAP message must NOT contain a DTD reference.
- A SOAP message must NOT contain XML Processing Instructions.

## Advantages

- SOAP is an integral part of the service-oriented architecture (SOA) and the Web services specifications associated with SOA. SOA without SOAP is difficult to imagine.
- SOAP allows the sender to create a message route based on the analytical services that have to be applied to the message on its way to the destination.
- SOAP is designed to support expansion, so it has all sorts of other acronyms and abbreviations associated with it, SOAP is highly extensible.
- It provides secure and compliant connections, controlling access, offers reliable delivery and failure recovery, and supports dynamic service discovery.

- The XML structure of SOAP is handy for applications that expect their information to be provided in XML form.
- SOAP can ride on various network protocols, including HTTP, which means it's quickly passed through firewalls, where other protocols might require unique accommodation.

## Disadvantages

- SOAP is human-readable and makes messages relatively more extensive than other protocols like CORBA and RPC, accommodating binary data..
- The most significant disadvantage of SOAP (and SOA overall) is its heavyweight protocol for a heavyweight architecture. The idea of passing a message through a string of nodes for processing mixes protocols and service bus architectural models for software. Neither of those two is considered optimal for microservice-based development that is popularly used.

## SOAP API

SOAP is almost always used in the context of a Web Services/SOA framework. The higher-level interface typically hides its application programming interface (API) for SOA. It is designed to break traditional monolithic applications into a multi-component, distributed form without losing security and control.

SOAP connects the elements of a complex set of distributed computing tools (the Web Services and SOA framework) and application components, which form a part of a comprehensive service-oriented framework.

SOAP defines more standards (like security and how messages are sent) than other APIs like REST, so it is complex. These might be deciding factors for organizations that require more comprehensive features in these standards. So, SOAP is rarely a good choice that applies to web services scenarios, which makes it ideal for enterprise-type situations.

The main reason to develop an application using SOAP API is its higher levels of security. For example, applications like reliable communication, ACID (Atomicity, Consistency, Isolation, Durability) compliance uses SOAP.

- Soap has high security. It has WS-Security, a built-in standard in addition to SSL support that gives enterprise-level features if you need them.
- Soap has successful/retry logic built-in for reliable messaging functionality even through SOAP intermediaries.
- SOAP has built-in ACID compliance that reduces anomalies and protects the integrity of the database (by prescribing how transactions and the database interact). ACID is used when handling financial or sensitive transactions as it is more conservative than other consistent models.

## Future of SOAP

SOAP was the first widely used protocol for connecting web services in an SOA (Service Oriented Architecture). Nowadays, nearly all modern development of distributed applications is based on RESTful principles. SOAP is almost confined to legacy applications and projects. Over time, its use is declining, as many other protocols and APIs have been introduced to overcome SOAP shortcomings and make it simple.