# SOAP Protocol

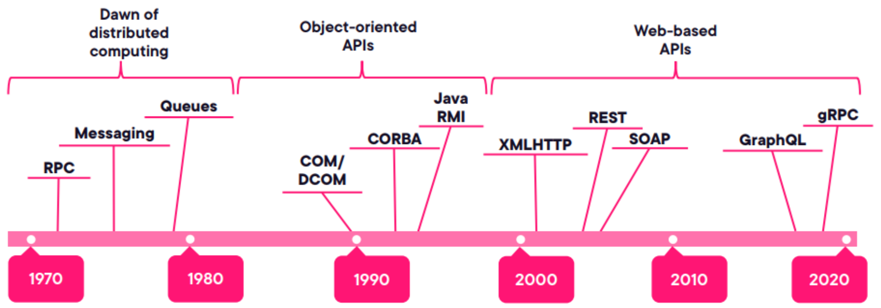
Ahmad Hamo
18-3-2-25

## Web APIs

- **Application Programming Interface (API)**
  - A particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another software program that implements that API.
  - Serves as an **interface** between different software programs and facilitates their interaction
- **Web API**
  - Typically, a defined set of HTTP request messages expressed in **SOAP or REST** along with a definition of the structure of response messages, typically expressed in **JSON or XML**.

1

## History



## History

- Web services evolved from previous technologies that served the same purpose such as **RPC**, Object Remote Procedure Call - ORPC (**DCOM, CORBA and JAVA RMI**).

- Web Services were intended to solve three main problems:

1. Interoperability
2. Firewall traversal
3. Complexity

## Interoperability

- Interoperability is the ability of different systems, devices, applications or products to connect and communicate in a coordinated way, without massive effort from the end user.
- Earlier distributed systems suffered from interoperability issues because each vendor implemented its own on-wire format for distributed object messaging.
- Development of **DCOM** apps strictly bound to **Windows** Operating system.
- Development of **RMI** bound to **Java** programming language.

## Firewall traversal

- Collaboration across corporations was an issue because distributed systems such as CORBA and DCOM used non-standard ports.
- CORBA: Typically uses port **900** for IIOP, but this can vary.
- DCOM: Uses port **135** for the Endpoint Mapper and dynamically assigned ports (1024-65535) for RPC communication.

- Web Services use HTTP as a transport protocol and most of the firewalls allow access though port **80** (HTTP), leading to easier and dynamic collaboration.

## Complexity

- Web Services are a developer-friendly service system: because they rely on widely adopted, standardized protocols like **HTTP**, **XML**, **JSON**, and **SOAP/REST**, which are familiar to most developers and easily integrated across platforms
- Most of the previously mentioned technologies such as RMI, DCOM, and CORBA involve a whole learning curve: require learning **specialized protocols**, **complex APIs**, and **platform-specific configurations.**
- Web Services' simplicity and platform independence make them more accessible for modern distributed systems development.

## What is a Web Service ?

Web service is a means by which **computers talk to each** other over the web using HTTP and other universally supported protocols.

**A Web service is an application that:**

- Exposed/running through a Web/application server
- Exposes Web methods to interested callers
- Listens for HTTP requests representing commands to invoke Web methods
- Executes Web methods and returns the results

## SOAP

- SOAP stands for "Simple Object Access Protocol" .
- Web Services expose useful functionality to Web users through a standard Web protocol called SOAP.
- SOAP is an XML vocabulary standard to enable programs on separate computers to **interact** across any network.
- SOAP is a simple markup language for **describing messages** between applications and how to implement the communication.
- SOAP uses **mainly HTTP** as a transport protocol. That is, HTTP message contains a SOAP message as its payload section.

## SOAP Characteristics

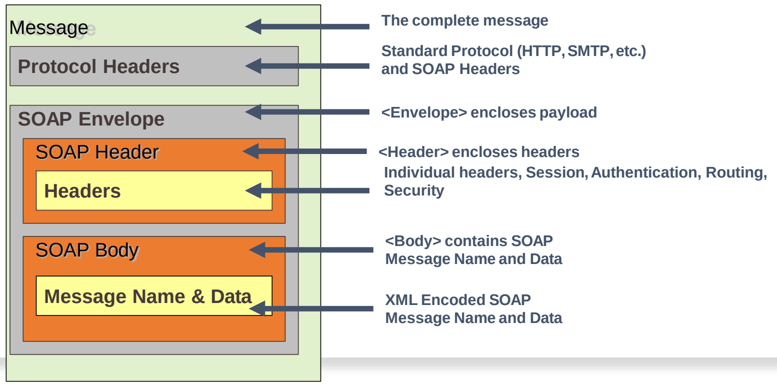
- SOAP has three major characteristics:
  - Extensibility – security and WS-routing.
  - Neutrality - SOAP can be used over any transport protocol such as HTTP, SMTP or even TCP.
  - Independent - SOAP allows for any programming model .

## SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

- A **required Envelope** element that identifies the XML document as a SOAP message.
- An **optional Header** element that contains header information.
- A **required Body** element that contains call and response information.
- An **optional Fault element** that provides information about errors that occurred while processing the message.

## What is a SOAP Message?


Message
Protocol Headers
SOAP Envelope
SOAP Header
Headers
SOAP Body
Message Name & Data
The complete message
Standard Protocol (HTTP, SMTP, etc.) and SOAP Headers
<Envelope> encloses payload
<Header> encloses headers
Individual headers, Session, Authentication, Routing, Security
<Body> contains SOAP Message Name and Data
XML Encoded SOAP Message Name and Data

## HTTP & SOAP Envelops

```
POST /orders HTTP/1.1
Host: restbucks.com
Content-Type: application/vnd.restbucks+xml
Content-Length: 32064

<order xmlns="http://..." .../>
```

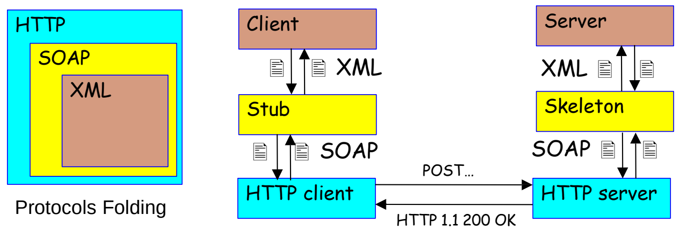
**HTTP envelope**

**SOAP envelope**

```
<soap:Envelope xmlns:soap="http://...">
  <soap:Header>
    <wsa:To xmlns:wsa="http://...">http://restbucks.com/orders</wsa:To>
  </soap:Header>
  <soap:Body>
    <order xmlns="http://..." .../>
  </soap:Body>
</soap:Envelope>
```

SOAP is a technology to support the exchange of XML-coded messages.

SOAP basic mechanism



Simple Object Access Protocol

Stub and skeleton both hide some complexity. **The stub** hides the serialization of parameters and the network-level communication in order to present a simple invocation mechanism to the caller. **The skeleton** is responsible for dispatching the call to the actual remote object implementation.

Serialization is the process of converting the state of an object into a form that can be persisted or transported. It allows us to transfer objects through a network by converting it into a byte stream.

## Stub

•The **stub** acts as a proxy for the remote service on the **client side**.
•It **hides the complexity** of:

- **Serialization**: Converting the method parameters into a SOAP message (XML format) that can be transmitted over the network.
- **Network Communication**: Handling the low-level details of sending the SOAP request to the server and receiving the SOAP response.

•Simplifies the client-side experience by handling serialization and network communication, making remote calls appear local.

•**Example**: A client calls a method on the stub, and the stub takes care of packaging the request, sending it to the server, and unpacking the response.

## Skeleton

**Skeleton (Server-Side)**
•The **skeleton** resides on the **server side** and acts as an intermediary between the SOAP message and the actual implementation of the remote service.
•It **hides the complexity** of:

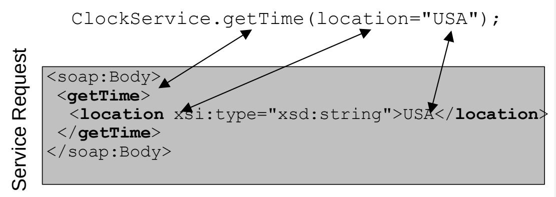
- **Deserialization**: Converting the incoming SOAP message (XML) into method parameters that the server-side implementation can understand.
- **Dispatching**: Forwarding the method call to the appropriate remote object implementation.

•Once the method is executed, the skeleton serializes the result into a SOAP response and sends it back to the client.
•Example: The skeleton receives a SOAP request, extracts the method name and parameters, invokes the corresponding method on the server-side object, and then packages the result into a SOAP response.

## SOAP Encoding

The rules:

- method name -> first level element in the SOAP Body
- arguments identifiers -> second level elements
- arguments values -> third level elements
- arguments types -> attribute xsi:type



```
ClockService.getTime(location="USA");
```

```
<soap:Body>
 <getTime>
  <location xsi:type="xsd:string">USA</location>
 </getTime>
</soap:Body>
```

## SOAP Request

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap+xml; charset=utf-8 Content-Length: 150

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle=http://www.w3.org/2001/12/soap- encoding”>

  <soap:Body xmlns:m="http://www.stock.org/stock">
      <m:GetStockPrice>
          <m:StockName>IBM</m:StockName>
      </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

## SOAP Response

```
HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: 126

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Body xmlns:m="http://www.stock.org/stock">
        <m:GetStockPriceResponse>
                <m:Price>34.5</m:Price>
        </m:GetStockPriceResponse>
    </soap:Body>
</soap:Envelope>
```

## SOAP Security

- SOAP uses HTTP as a transport protocol and hence can use HTTP security mainly HTTP over SSL.
- The WS-Security specification defines a complete encryption system.

## WSDL

- Web Services describe
  - what they are
  - where they can be found
  - how they should be used

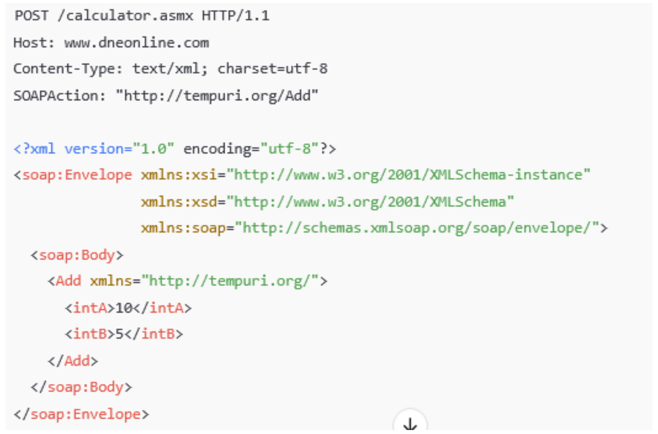
## WSDL

- WSDL stands for Web Services Description Language.
- WSDL is an XML vocabulary for **describing** Web services. It allows developers to **describe Web Services and their capabilities, in a standard manner**.
- WSDL specifies what a **request message must contain and what the response message** will look like in clear notation. In other words, **it is a contract between the web service and the client who wishes to use this service.**
- In addition to describing message contents, WSDL **defines where the service is available and what communications protocol is used to talk to the service.**

## The WSDL Document Structure

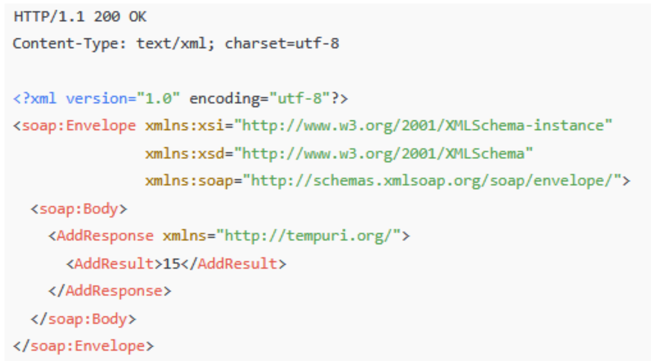
- A WSDL document is just a simple XML document.
- It defines a web service using these major elements:
  - **types -** The data types used by the web service.
  - **element** – structure and data validation.
  - **message -** define request/response for a service..
  - **port type** - The operations performed by the web service.
  - **binding-** The communication protocols used by the web service.
  - **Service -** The location of the service.

Sample SOAP Request for Add Operation

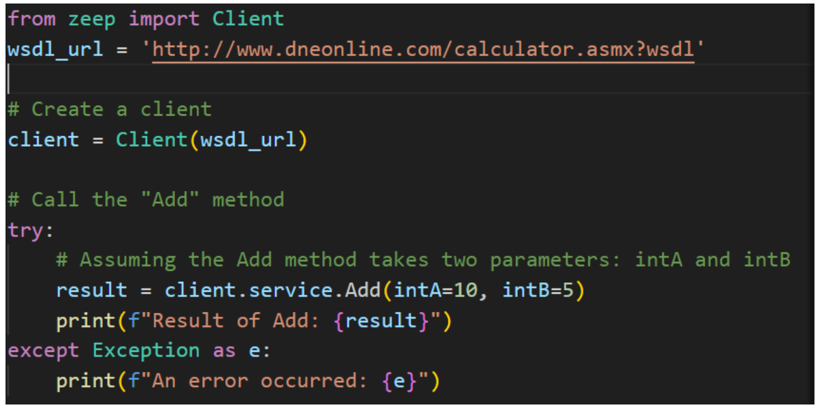


```
POST /calculator.asmx HTTP/1.1
Host: www.dneonline.com
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tempuri.org/Add"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Add xmlns="http://tempuri.org/">
      <intA>10</intA>
      <intB>5</intB>
    </Add>
  </soap:Body>
</soap:Envelope>
```

## Example: Client code - Python

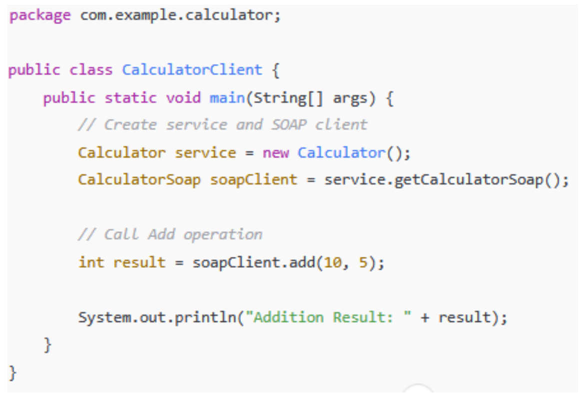


```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AddResponse xmlns="http://tempuri.org/">
      <AddResult>15</AddResult>
    </AddResponse>
  </soap:Body>
</soap:Envelope>
```



```python
from zeep import Client
wsdl_url = 'http://www.dneonline.com/calculator.asmx?wsdl'

# Create a client
client = Client(wsdl_url)

# Call the "Add" method
try:
    # Assuming the Add method takes two parameters: intA and intB
    result = client.service.Add(intA=10, intB=5)
    print(f"Result of Add: {result}")
except Exception as e:
    print(f"An error occurred: {e}")
```

## Example: Client code - Java



```
package com.example.calculator;

public class CalculatorClient {
    public static void main(String[] args) {
        // Create service and SOAP client
        Calculator service = new Calculator();
        CalculatorSoap soapClient = service.getCalculatorSoap();

        // Call Add operation
        int result = soapClient.add(10, 5);

        System.out.println("Addition Result: " + result);
    }
}
```

Run this command to generate the Java stubs from the WSDL:
**wsimport -keep -p com.example.calculator http://www.dneonline.com/calculator.asmx?WSDL**
**jaxws-ri-2.3.1.zip package**
This creates Java classes inside com.example.calculator.

## The WSDL Document Structure

Let's see an example!

Structure of a WSDL Document (oracle.com)

https://www.w3schools.com/xml/tempconvert.asmx?WSDL

## Sample WSDLs

[Public SOAP APIs (getpostman.com)]

www.dneonline.com/calculator.asmx?wsdl (Demo)

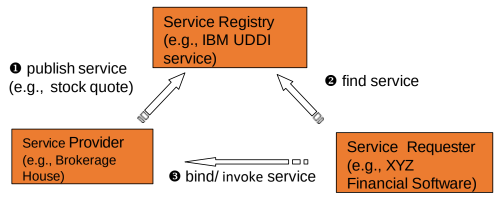
https://www.w3schools.com/xml/tempconvert.asmx

```
<message name="GetStockPriceRequest">
     <part name="stock" type="xs:string"/>
 </message>
<message name="GetStockPriceResponse">
     <part name="value" type="xs:string"/>
 </message>

<portType name="StocksRates">
      <operation name="GetStockPrice">
            <input message="GetStockPriceRequest"/>
            <output message="GetStockPriceResponse"/>
      </operation>
</portType>
```

WSDL Document

## Web Services Model

The Web Services model follows the *publish*, *find*, and *bind* paradigm.


Service Registry (e.g., IBM UDDI service)
❶ publish service (e.g., stock quote)
❷ find service
Service Provider (e.g., Brokerage House)
❸ bind/ invoke service
Service Requester (e.g., XYZ Financial Software)

Life Cycle of a Web Service Execution
(Registry, Lookup, and Consumption)

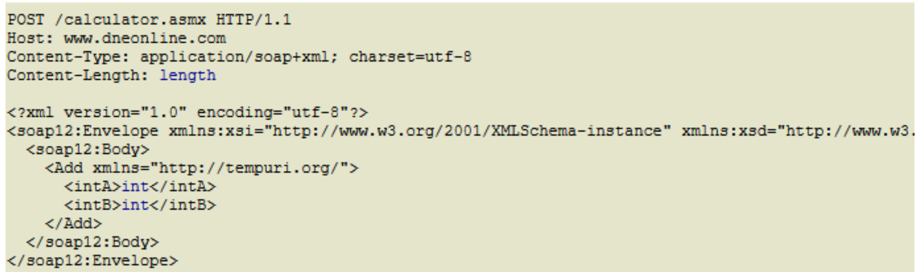
## UDDI

- Universal Description, Discovery, and Integration
- Enable companies find publicly available Web Services on the Internet or corporate Intranets.
- UDDI is a directory for storing information about web services , like yellow pages.
- UDDI is a directory of web service interfaces described by WSDL.

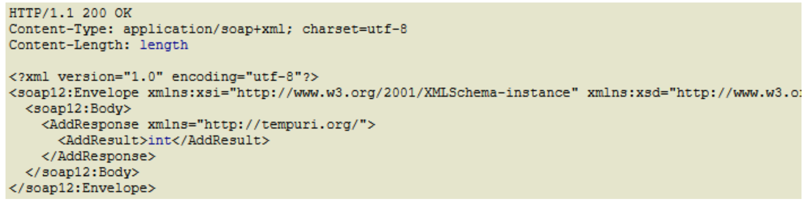
## References

- RESTful Web APIs: Services for a Changing World, By Leonard Richardson, Mike Amundsen, Sam Ruby 2020.
- https://www.w3.org/
- https://developer.mozilla.org/
- Web server vs. Application server (educative.io)
- The Next Dimension of Enterprise Computing, Dr. Billy B. L. Lim, School of Information Technology, Illinois State University

## Example: SOAP Request & Response



```
POST /calculator.asmx HTTP/1.1
Host: www.dneonline.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.
  <soap12:Body>
    <Add xmlns="http://tempuri.org/">
      <intA>int</intA>
      <intB>int</intB>
    </Add>
  </soap12:Body>
</soap12:Envelope>
```



```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.o
  <soap12:Body>
    <AddResponse xmlns="http://tempuri.org/">
      <AddResult>int</AddResult>
    </AddResponse>
  </soap12:Body>
</soap12:Envelope>
```

## Example: WSDL

```
<s:element name="Add">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="intA" type="s:int"/>
          <s:element minOccurs="1" maxOccurs="1" name="intB" type="s:int"/>
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="AddResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="AddResult" type="s:int"/>
        </s:sequence>
      </s:complexType>
    </s:element>
```

## Example: WSDL

```
<wsdl:message name="AddSoapIn">
   <wsdl:part name="parameters" element="tns:Add"/>
</wsdl:message>
<wsdl:message name="AddSoapOut">
   <wsdl:part name="parameters" element="tns:AddResponse"/>
</wsdl:message>


<wsdl:portType name="CalculatorSoap">
    <wsdl:operation name="Add">
      <wsdl:input message="tns:AddSoapIn"/>
      <wsdl:output message="tns:AddSoapOut"/>
    </wsdl:operation>
.....
</wsdl:portType>
```

## Example2: Mapping WSDL --> Code

```
<wsdl:portType name="ordering">
  <wsdl:operation name="placeOrder">
    <wsdl:input message="restbucks:Order"/>
    <wsdl:output message="restbucks:OrderConfirmation"/>
    <wsdl:fault name="fault" message="restbucks:OrderException"/>
  </wsdl:operation>
...
</wsdl:portType>
```

```
public class OrderingService {
  public OrderConfirmation placeOrder(Order order)
                                    throws OrderException {
    ...
  }
```