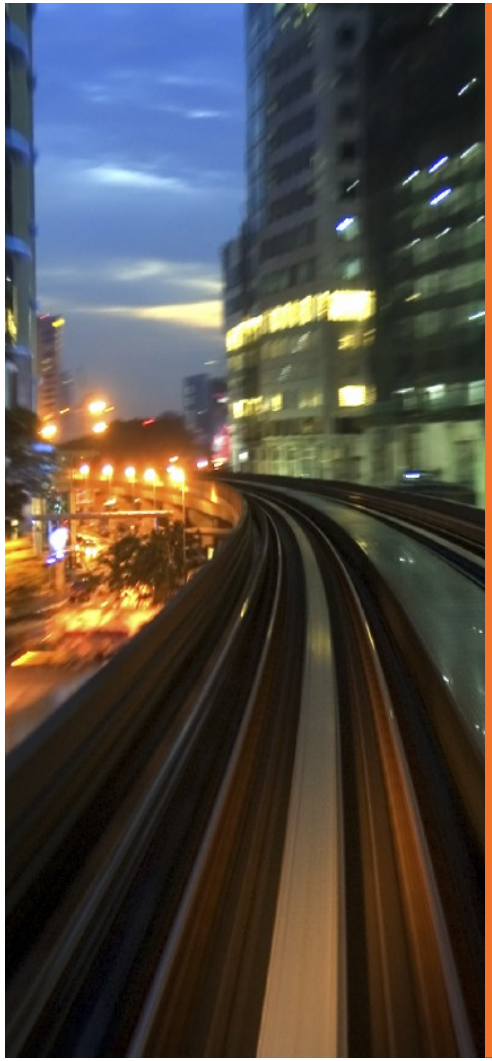
aeturnum
time to turn to innovative solutions

Introduction to Web Service

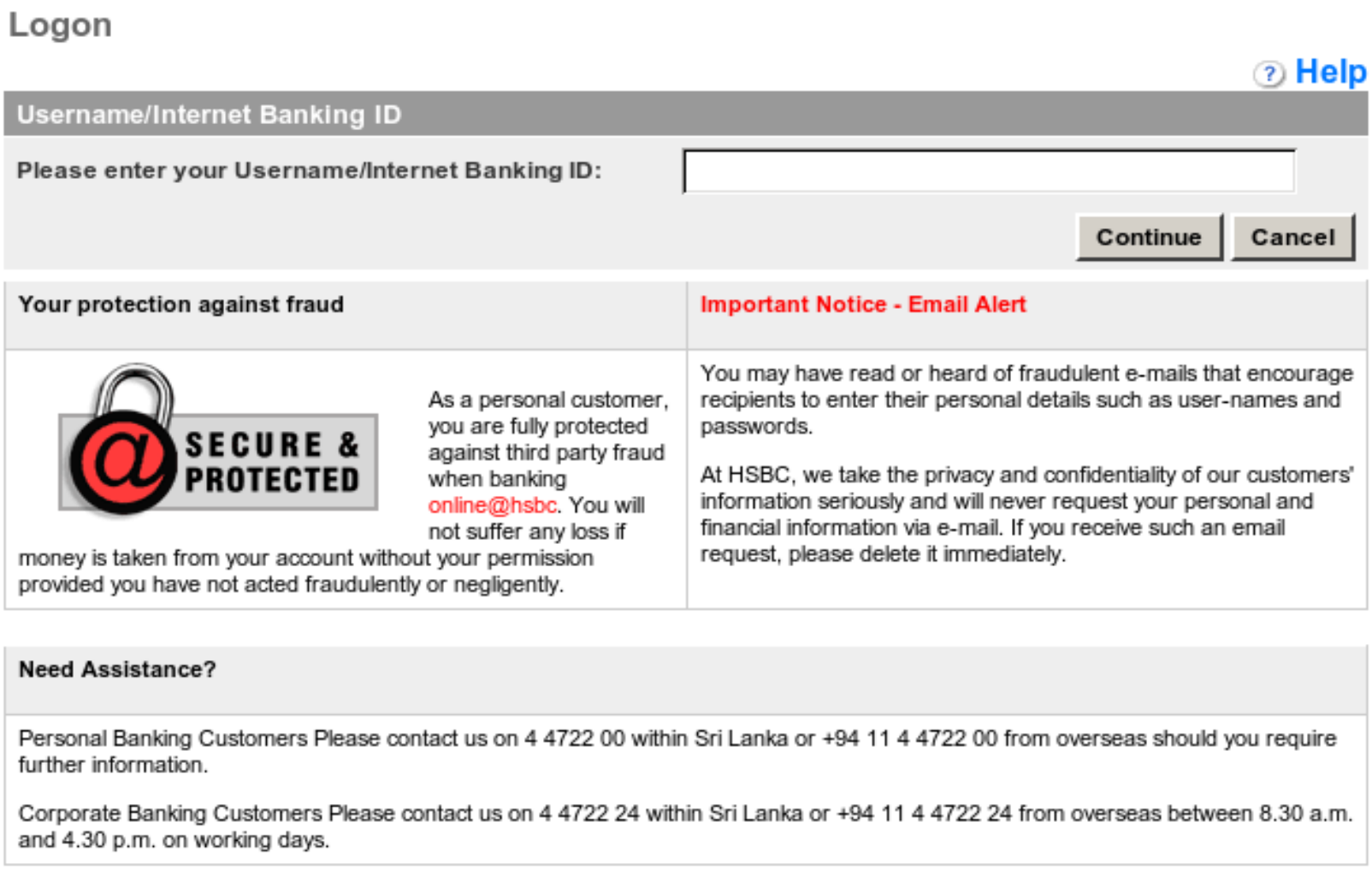
Sagara Gunathunga
(Apache web Service and Axis committer)

CONTENTS



Why you need
Web Services
?

How do you interact with on-line financial service?



Logon

Help

Username/Internet Banking ID

Please enter your Username/Internet Banking ID:

Continue Cancel

Your protection against fraud

SECURE & PROTECTED

As a personal customer, you are fully protected against third party fraud when banking online@hsbc. You will not suffer any loss if money is taken from your account without your permission provided you have not acted fraudulently or negligently.

Important Notice - Email Alert

You may have read or heard of fraudulent e-mails that encourage recipients to enter their personal details such as user-names and passwords.

At HSBC, we take the privacy and confidentiality of our customers' information seriously and will never request your personal and financial information via e-mail. If you receive such an email request, please delete it immediately.

Need Assistance?

Personal Banking Customers Please contact us on 4 4722 00 within Sri Lanka or +94 11 4 4722 00 from overseas should you require further information.

Corporate Banking Customers Please contact us on 4 4722 24 within Sri Lanka or +94 11 4 4722 24 from overseas between 8.30 a.m. and 4.30 p.m. on working days.

Conclusion

Web based interfaces are ideal for human interaction , not for machine interaction.

In most of the real world scenarios it is difficult to implement and maintain machine readable web interfaces.

How do you interact with on-line financial service?

How do you interact with on-line financial service?

Conclusion

1.) Need a protocol to define machine readable, platform independent and language independent contract (Interface).

2.) Need a language independent, platform independent, transport independent messaging format (protocol).

3.) Need a protocol to publish services into a repository and mechanism to query those services.

What are Web services?

Web services are loosely coupled, contracted components that communicate via XML-based interfaces.

- loosely coupled: - they can be changed independently
- platform independent
- contracted: in and output are publicly available
- components: interface encapsulates the code
- XML-based interfaces: - human readable
- firewall friendly
- self-describing (allows for discovery of their functionality)

Web Service Characteristics

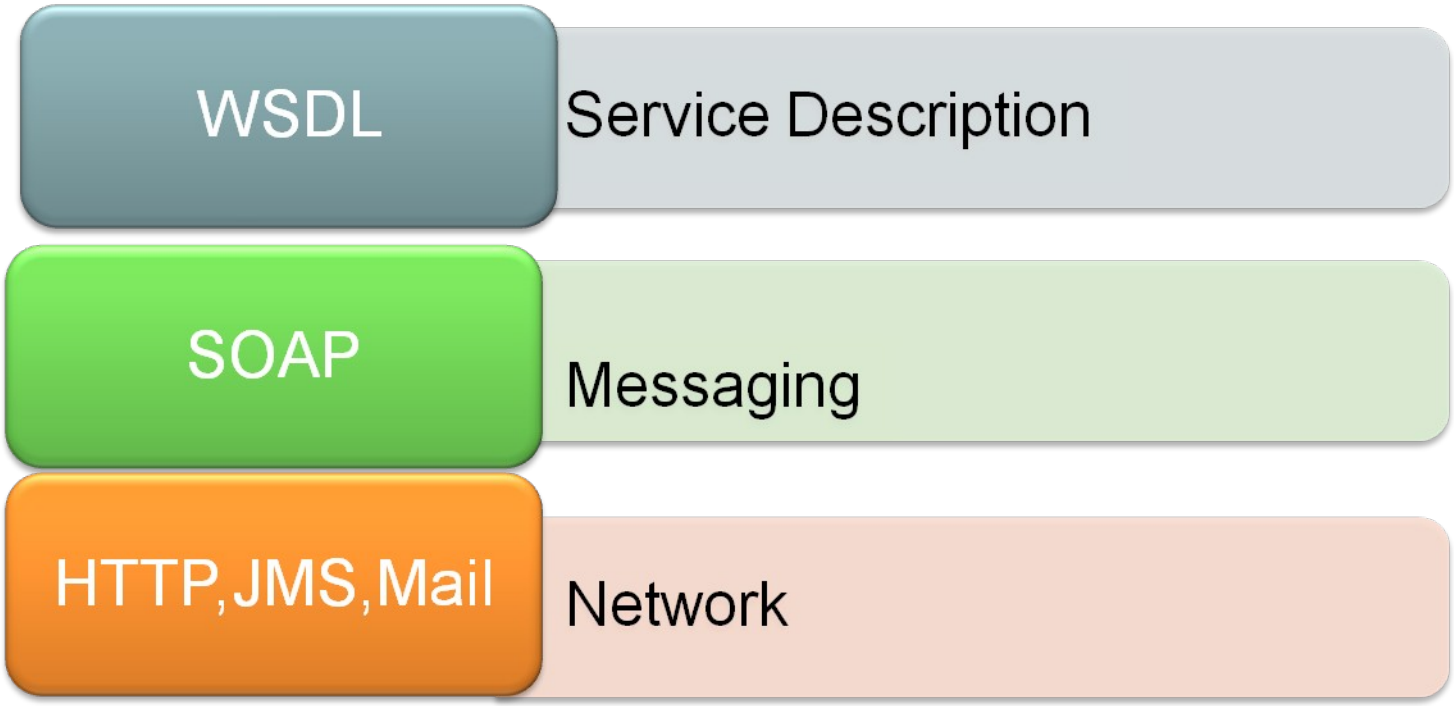
1.) Network accessible via standardized XML messaging, typically based on HTTP.

2.) Described using a standard, formal XML notion (service description) Covers all the details, necessary to invoke the service (message formats, location, etc.)

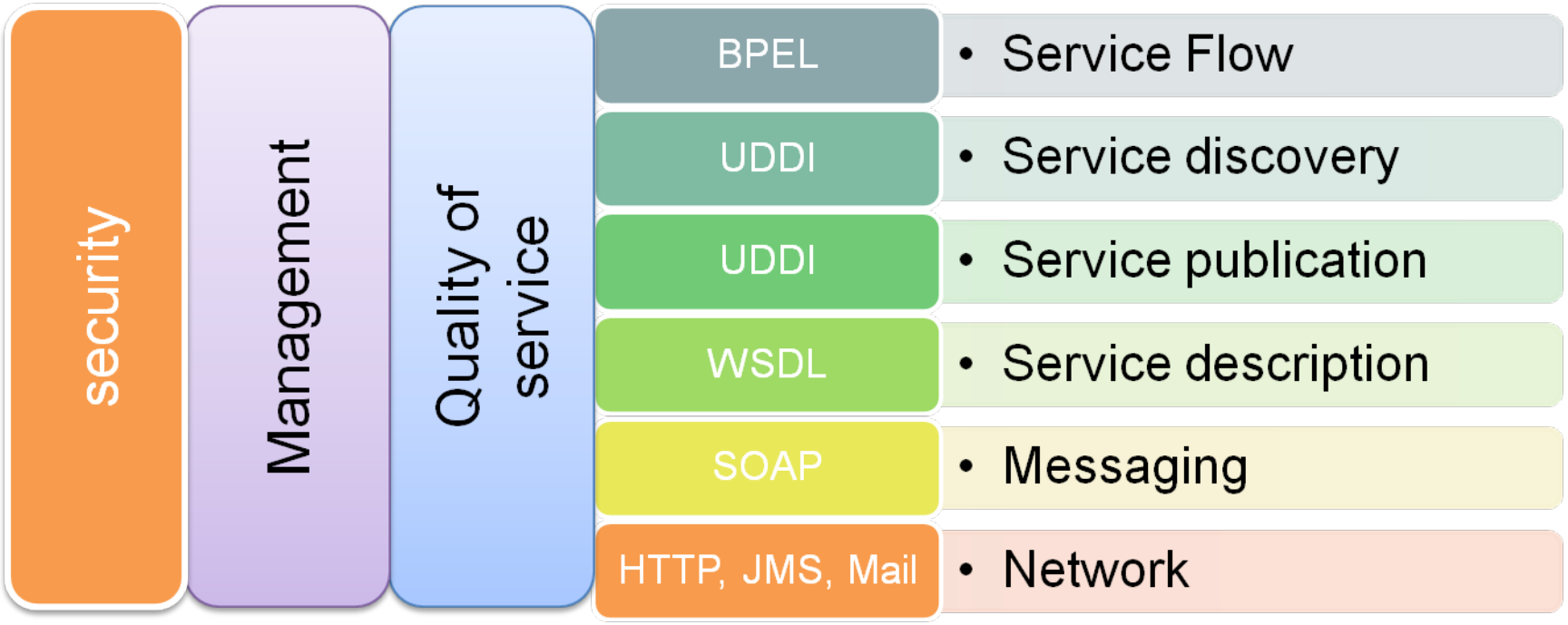
3.) Interface hides the implementation details.

4.)Supports Security , reliable messaging, Notification and QoS.

Web Services Stack

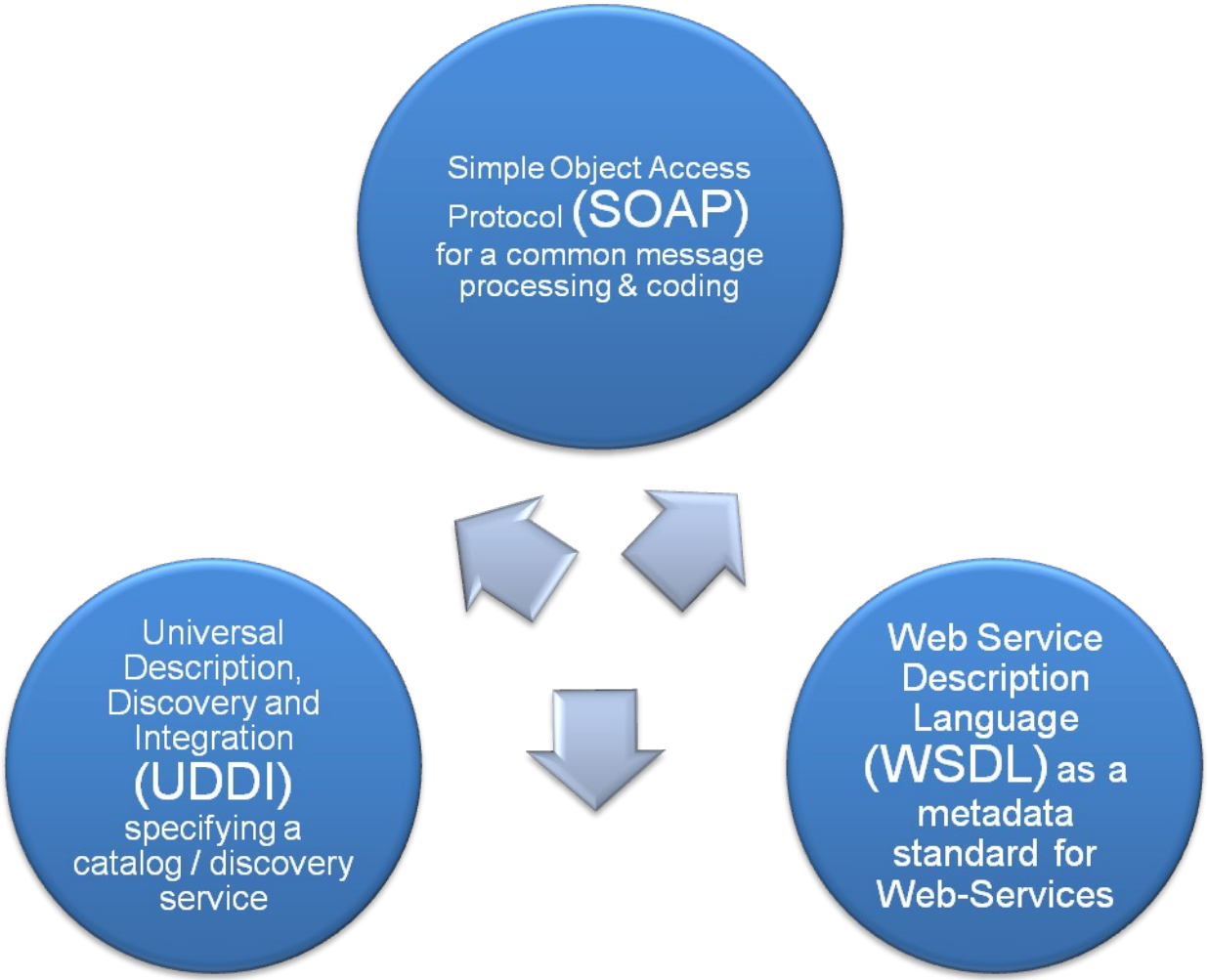


Web Services Stack



Web Services Specification

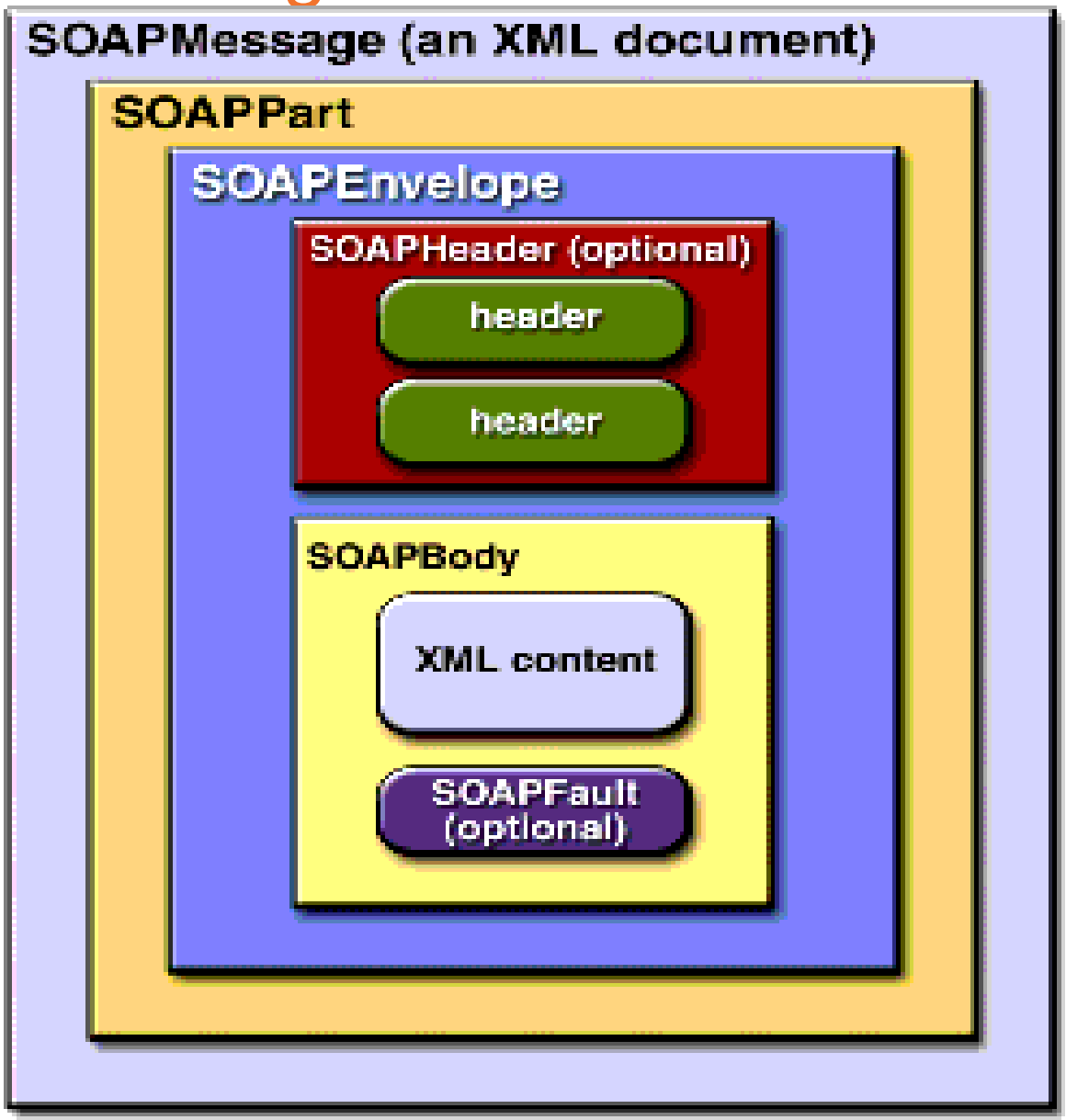
There are three basic specifications, forming the basis for state-of-the-art Web-Service technology



Simple Object Access Protocol (SOAP)

SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. SOAP uses XML technologies to define an extensible messaging framework, which provides a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation-specific semantics.

SOAP message structure



SOAP ..

- SOAP originally stood 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.
-
- Soap uses mainly HTTP as a transport protocol. That is, HTTP message contains a SOAP message as its payload section.

SOAP ..

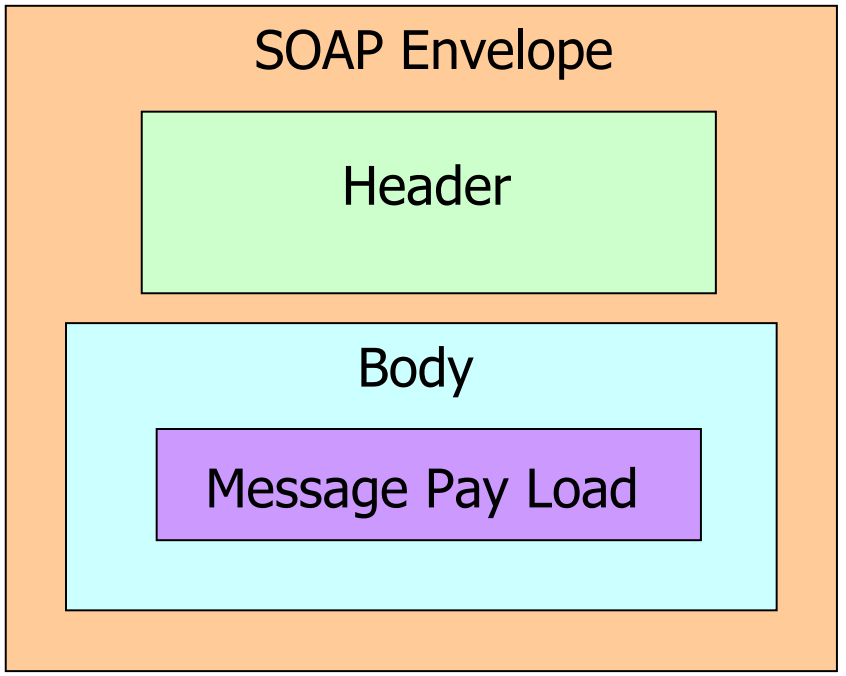
A SOAP message is an ordinary XML document containing the following elements:

- 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.

SOAP over HTTP

HTTP Headers

```
<soap:Envelope>
      <soap:Header>
            XML…
      </soap:Header>
      <soap:Body>
            XML payload…
       </soap:Body>
</soap:Envelope>
```



Example - 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>
```

Web Services Description Language (WSDL)

- WSDL stands for Web Services Description Language.

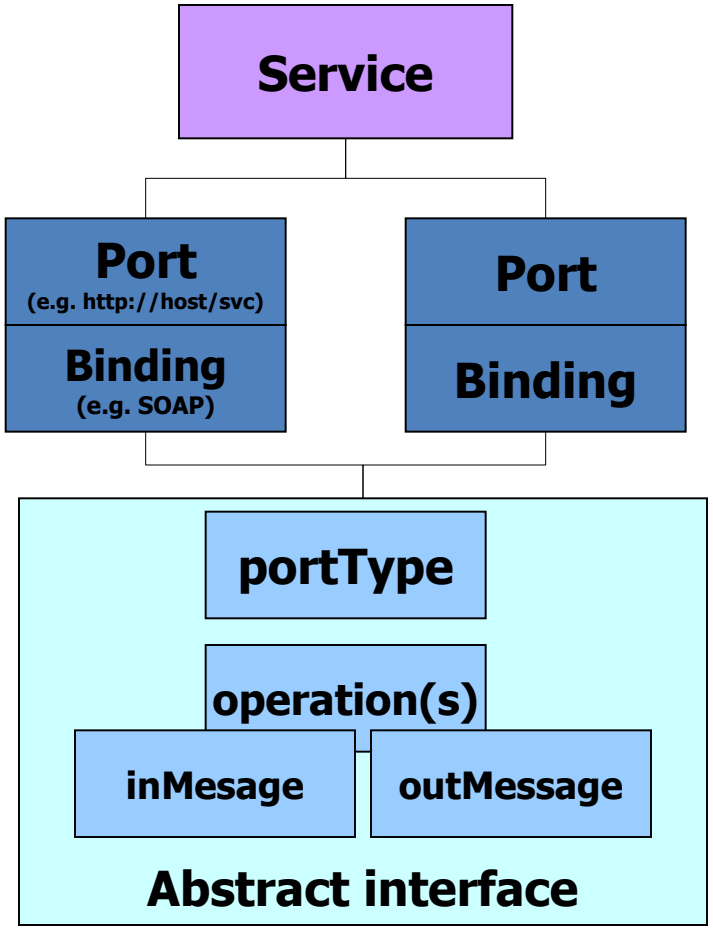
- WSDL is an XML vocabulary for describing Web
- services. It allows developers to describe Web Services and their capabilities, in a standard manner.

- WSDL specifies what a request message must contain and what the response message will look like in unambiguous notation. In other words, it is a contract between the XML 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.

WSDL Structure

- portType
  - Abstract definition of a service (set of operations)
- Multiple bindings per portType:
  - How to access it
  - SOAP, JMS, direct call
- Ports
  - Where to access it



The WSDL Document Structure

- A WSDL document is just a simple XML document.
- It defines a web service using these major elements:
  - **port type** - The operations performed by the web service.
  - **message -** The messages used by the web service.
  - **types -** The data types used by the web service.
  - **binding** **-** The communication protocols used by the web service.

Example WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions>
        <wsdl:message name="execLocalCommandResponse">
        <wsdl:message name="execLocalCommandRequest">
        <wsdl:portType name="SJwsImp">
            <wsdl:operation name="execLocalCommand" parameterOrder="in0">
                <wsdl:input message="impl:execLocalCommandRequest"
                                    name="execLocalCommandRequest"/>
                <wsdl:output message="impl:execLocalCommandResponse"
                                    name="execLocalCommandResponse"/>
            </wsdl:operation>
        </wsdl:portType>
        <wsdl:binding name="SubmitjobSoapBinding" type="impl:SJwsImp">
            <wsdlsoap:binding style="rpc"
                                transport="http://schemas.xmlsoap.org/soap/http"/>
                <wsdl:operation name="execLocalCommand">
                        <wsdlsoap:operation soapAction=""/>
                        <wsdl:input name="execLocalCommandRequest">
                        <wsdl:output name="execLocalCommandResponse">
                </wsdl:operation>
            </wsdl:binding>
        <wsdl:service name="SJwsImpService">
            <wsdl:port binding="impl:SubmitjobSoapBinding" name="Submitjob">
        </wsdl:service>
</wsdl:definitions>
```

Example: WSDL *types* for Custom Data Definition

```
<wsdl:types>
    <schema targetNamespace="http://.../GCWS/services/Submitjob"
            xmlns:impl="http://.../GCWS/services/Submitjob"
            xmlns="http://www.w3.org/2001/XMLSchema">

      <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />

      <complexType name="ArrayOf_xsd_string">
          <complexContent>
              <restriction base="soapenc:Array">
                  <attribute ref="soapenc:arrayType"
                             wsdl:arrayType="xsd:string[]" />
              </restriction>
          </complexContent>
      </complexType>

      <element name="ArrayOf_xsd_string" nillable="true"
                         type="impl:ArrayOf_xsd_string" />
    </schema>
</wsdl:types>
```

Example - message

```
<wsdl:message
name="execLocalCommandResponse">
        <wsdl:part
                        name="execLocalCommandReturn"
                         type="impl:ArrayOf_xsd_string" />
 </wsdl:message>

<wsdl:message              name="execLocalCommandRequest">
        <wsdl:part name="in0" type="xsd:string" />
</wsdl:message>
```

WSDL - portTypes

- *portType* elements map messages to *operations*.
- You can think of portType==class, operation==class methods.
- Operations can contain input, output, and/or fault bindings for messages.
- An operation may support of the following message styles:
  - One-way: request only
  - Two-way: request/response
  - Solicit-response: server “push” and client response
  - Notification: one-way server push

```
<wsdl:portType name="SJwsImp">

    <wsdl:operation         name="execLocalCommand"
                            parameterOrder="in0">
              <wsdl:input
                        message="impl:execLocalCommandRequest"
                        name="execLocalCommandRequest" />

             <wsdl:output
                       message="impl:execLocalCommandResponse"
                       name="execLocalCommandResponse" />
   </wsdl:operation>

</wsdl:portType>
```

Service and Port Definitions

- So far, we have defined the class method interfaces (portTypes) and the rules for binding to a particular protocol.
- *Port* elements define how the bindings (and thus the portTypes) are associated with a particular server.
- The *service* element collects *ports.*

```
<wsdl:service name="SJwsImpService">

    <wsdl:port
                    binding="impl:SubmitjobSoapBinding"
                   name="Submitjob">
        <wsdlsoap:address
                   location="http://.../GCWS/services/Submitjob" />
   </wsdl:port>

</wsdl:service>
```

PortType Bindings

- portTypes are abstract interface definitions.
  - Don’t say anything about how to invoke a remote method.
- Remote invocations are defined in *binding* elements.
- Binding elements are really just place holders that are extended for specific protocols
  - WSDL spec provides SOAP, HTTP GET/POST, and MIME extension schema examples.

```
<wsdl:binding
                name="SubmitjobSoapBinding" type="impl:SJwsImp">

    <wsdlsoap:binding style="rpc"
                transport="http://schemas.xmlsoap.org/soap/http" />

    <wsdl:operation name="execLocalCommand“>
       <wsdlsoap:operation soapAction="" />
        <wsdl:input name="execLocalCommandRequest">
            <wsdlsoap:body
                 encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
                 namespace="http://.../GCWS/services/Submitjob"
                 use="encoded" />
        </wsdl:input>
        <wsdl:output name="execLocalCommandResponse">
            <wsdlsoap:body
                encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
                namespace=http://.../GCWS/services/Submitjob
                 use="encoded" />
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
```

Web Service frameworks (Java)

Apache Axis2

- http://ws.apache.org/axis2/

Apache CXF

- http://cxf.apache.org/

Sun Metro

- https://metro.dev.java.net/

Spring WS

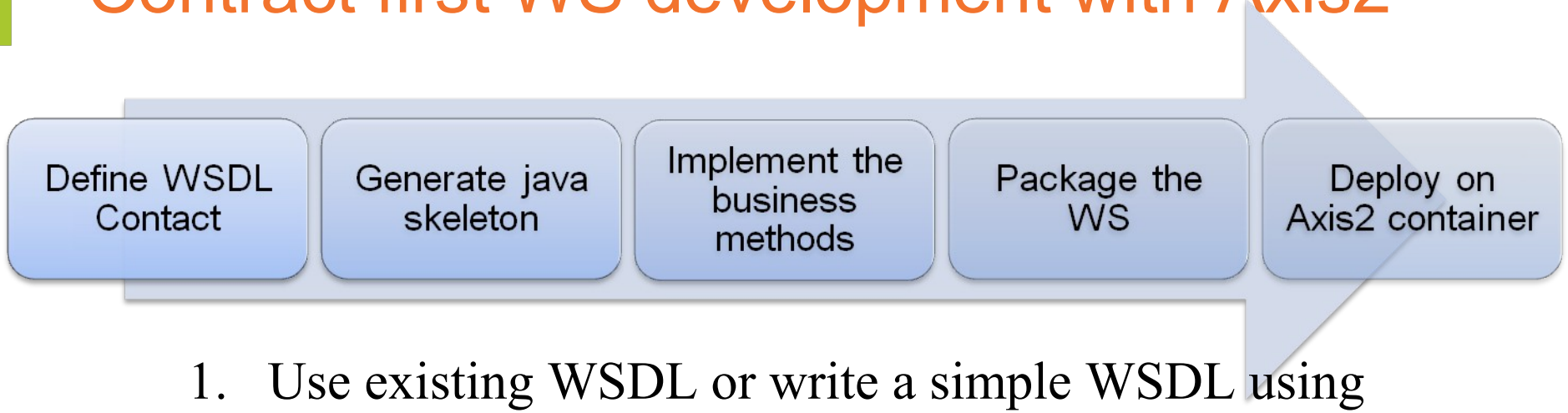
- http://static.springsource.org/spring-ws/sites/1.5/

Contract first WS development

1. Start with existing WSDL document or writing a new WSDL document, all of the web service frameworks support for contact first web service development.

2. Both Apache Axis2 and CXF provide ‘WSDL2Java” tool , that capable to generate source codes from WSDL contract. Spring -WS doesn't provide any code generation tool but supports for Contact first approach .

Contract first WS development with Axis2



1. Use existing WSDL or write a simple WSDL using a XML editor or Eclipse WSDL editor.
2. Generate java skeleton using “WSDL2JAVA” tool.
3. Complete server side methods .
4. Package as .AAR (with the generated service.xml)
5. Deploy in to “services” directory of Axis2 installation.

Linux

- sh wsdl2java.sh -uri sample1.wsdl -ss –sd

Windows

- wsdl2java.bat -uri sample1.wsdl -ss -sd

Code first WS development

1. Start with existing codes , most of the time people use this approach when they want to expose existing business features as a web service.

2. Most of the Web service framework supports for automatic WSDL generation from services. (Axis2 , CXF)

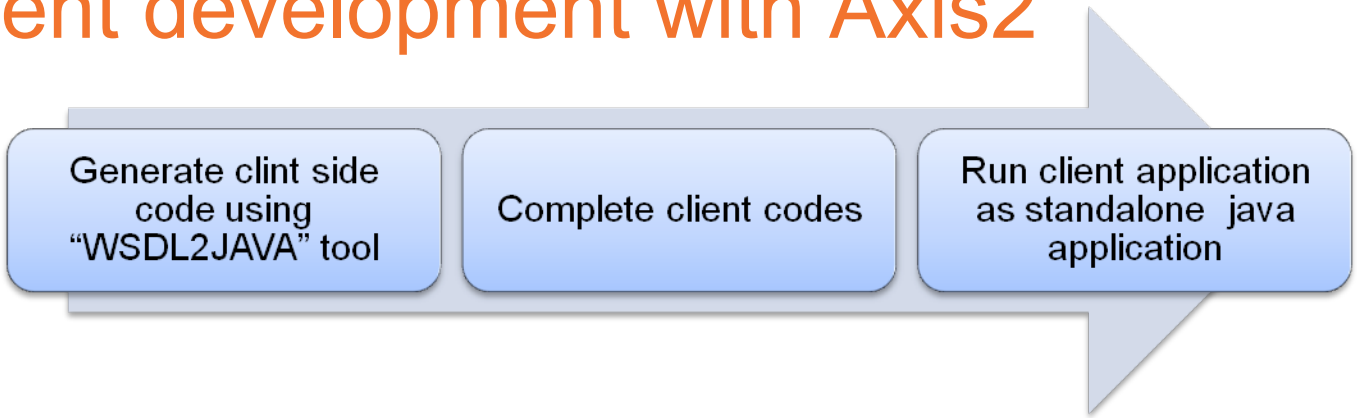
3. Users can utilize this automatic code generation feature or they can disable this feature to assign existing WSDL file.

Code first WS development with Axis2

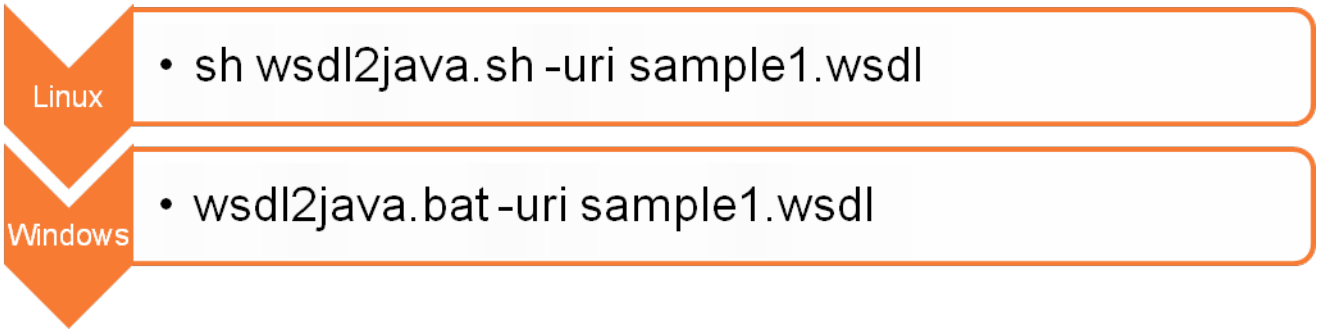
Optionally you can write a interface too, specially with Spring, Hibernate and AOP related services.

Service .xml - is a Axis2 specific service description file that contains important meta data about the service such as service class , service interface , message receivers etc.

WS client development with Axis2



Once you have a service description for a service you can generate WS client code using a tools like "WSDL2JAVA" , the generated code contain the required logic to serialize messages, service location , QoS attributes etc.



References

- This slide and source codes for samples.
  - http://people.apache.org/~sagara/ws/intro10

- http://ws.apache.org/axis2/
- http://wso2.org/library/95
- http://www.developer.com/services/article.php/3613896/Writing-ar
- http://wso2.org/library/2873

Thank You

Aeturnum Lanka (Pvt) Ltd

197, Stanley Thilakaratna Mw, Nugegoda 10250, Sri Lanka

Phone: +94 11 5518177 | **Email:** info@aeturnum.com

Web: www.aeturnum.com | www.athiya.com

aeturnum
time to turn to innovative solutions