4

SOAP Web Services: Overview

Objectives

After completing this lesson, you should be able to do the following:

- Describe the basic structure of a Simple Object Access Protocol (SOAP) message and how it is encapsulated by transports
- Explain how WSDL defines a web service, including its message representation and transport mechanism
- Explain the purpose of WS-I Basic Profile and WS-Policy

Course Roadmap

Application Development Using Webservices [SOAP and Restful]

- Lesson 1: Introduction to Web Services
- Lesson 2: Creating XML Documents
- Lesson 3: Processing XML with JAXB
- **Lesson 4: SOAP Web Services Overview** — **You are here!**
- Lesson 5: Creating JAX-WS Clients

Course Roadmap

Application Development Using Webservices [SOAP and Restful]

- Lesson 6: Exploring REST Services
- Lesson 7: Creating REST Clients
- Lesson 8: Bottom Up JAX Web Services
- Lesson 9: Top Down JAX Web Services
- Lesson 10: Implementing JAX RS Web Services

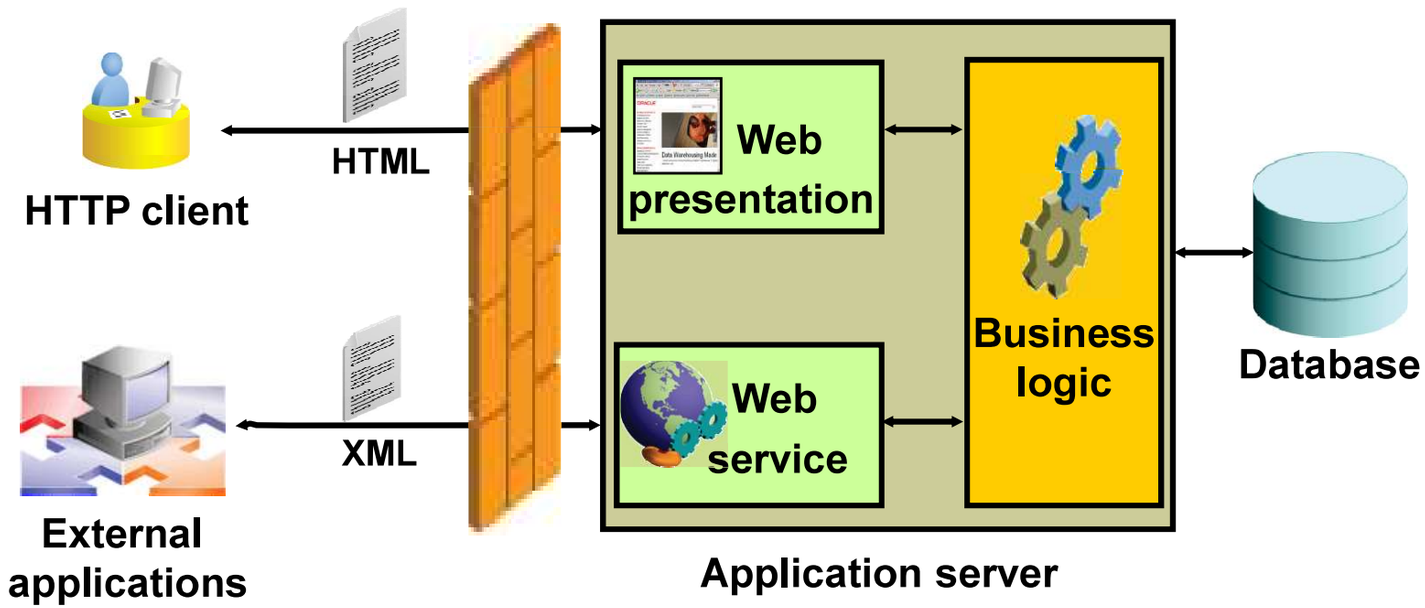
Course Roadmap

Application Development Using Webservices [SOAP and Restful]

- Lesson 11: Web Service Error Handling
- Lesson 12: Java EE Security and Securing JAX WS

Web Services

- Is a technology that is based on a set of standards for building interoperable distributed applications
- Performs self-describing business functions



Reasons for Using SOAP

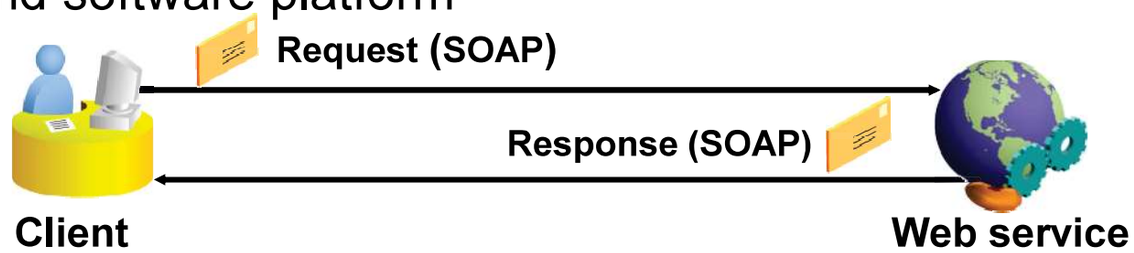
The SOAP web services specification sets out to define an interoperable, platform-independent means for component interaction. SOAP web service requirements include:

- Decouple message representation from transport mechanisms
- Support extensible frameworks

SOAP: XML Messaging for Web Services

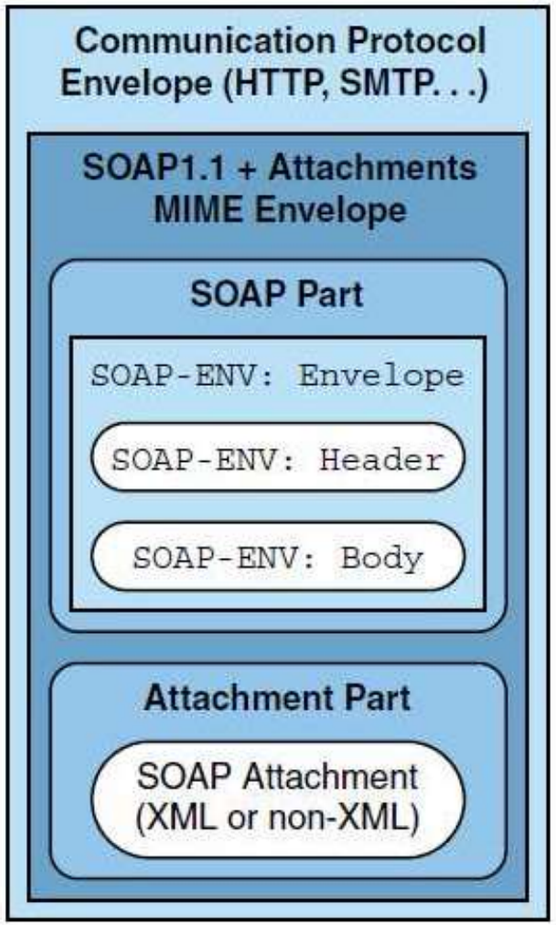
SOAP:

- Is an XML-based protocol for exchanging data
- Represents requests and responses as XML messages
- Uses HTTP and other protocols at the transport layer
- Supports data encoding and literal styles
- Hides details of implementations
- Works with:
  - Any programming language
  - Any hardware and software platform

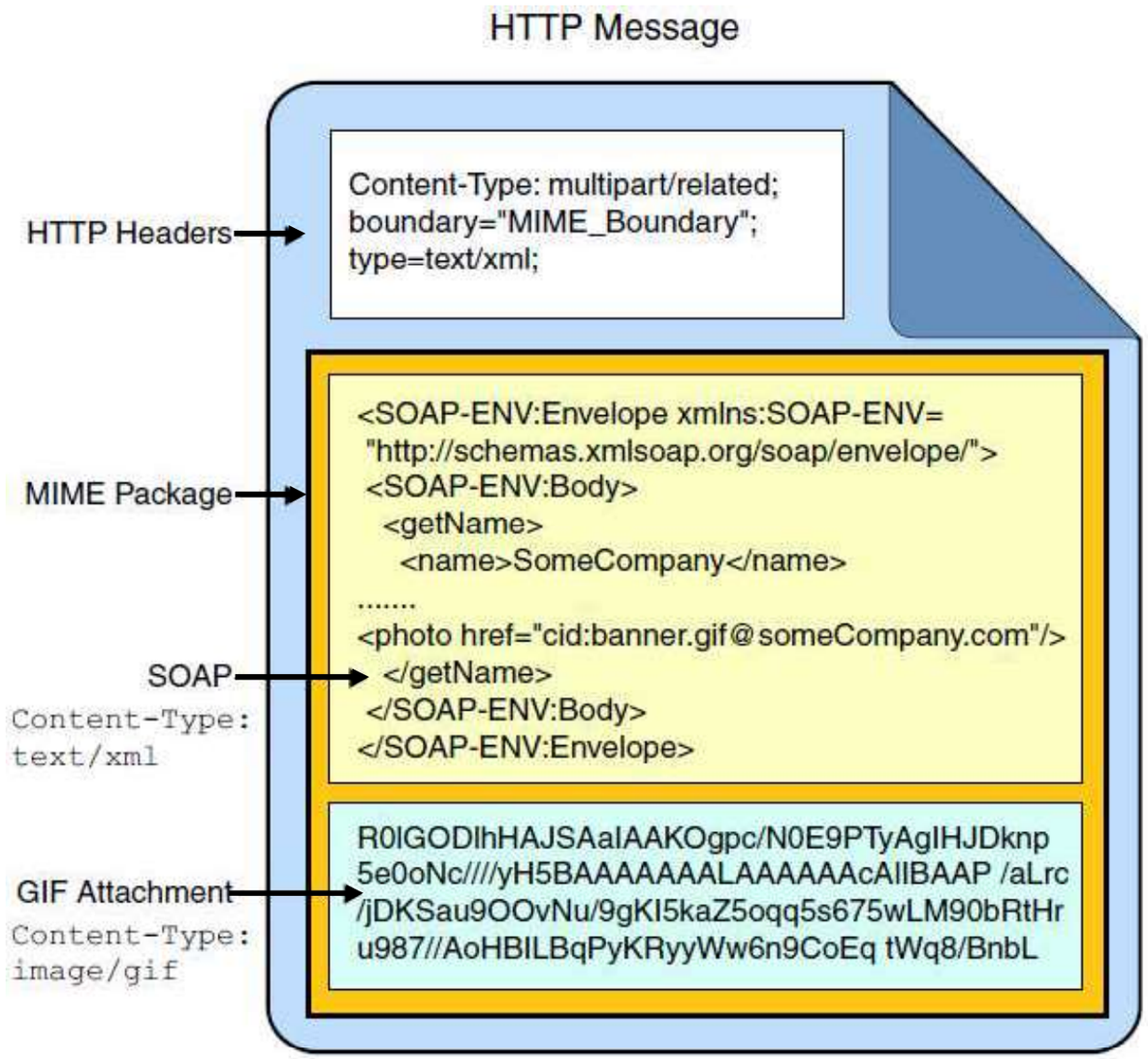


Extensible Message Representation

Simple Object Access Protocol



SOAP Over HTTP



Raw SOAP/HTTP Request

```
Accept: text/html, image/gif, image/jpeg, */*; q=.2
Connection: Keep-Alive
Content-Length: 206
Content-Type: text/xml; charset=utf-8
Host: localhost:7001
SOAPAction: ""
User-Agent: Oracle JAX-RPC 1.1
<env:Envelope
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<env:Body>
  <getHello xmlns="http://ou/">
    <arg0 xmlns="">matt</arg0>
  </getHello>
</env:Body>
</env:Envelope>
```

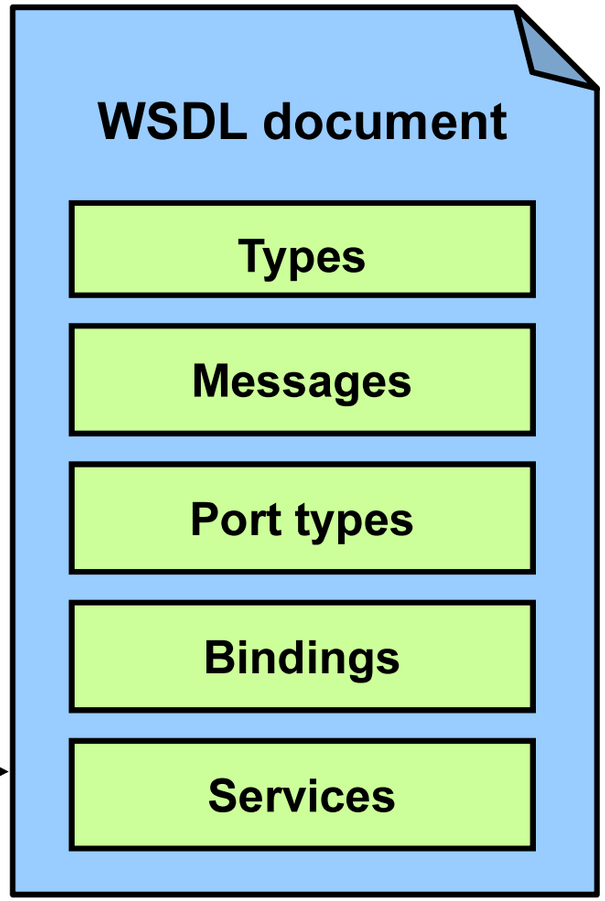
Raw SOAP/HTTP Response

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

<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope
   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
  <ns2:getHelloResponse xmlns:ns2="http://ou/">
    <return>Hello matt!</return>
  </ns2:getHelloResponse>
</S:Body>
</S:Envelope>
```

Web Services Description Language (WSDL)

- A WSDL document is an XML document that describes:
  - What the service does
  - How the service is accessed
  - Where the service is located
- It defines the messages and the operations of a service abstrac



WSDL document structure

Structure of a WSDL File

<definitions>: Root WSDL Element

<types>: What data types will be transmitted?

<message>: What exact information is expected?

<portType>: What operations (functions) will be supported?

<binding>: How will the messages be transmitted on the wire?
What SOAP-specific details are there?

<service>: Define the collection of ports that make up the service and where is the service located?

WSDL: Sample Structure

```
<?xml version='1.0' encoding='UTF-8'?>
<definitions xmlns:wsu="http://docs.oasis-
   open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
   1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
   xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
   xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://ou/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   targetNamespace="http://ou/" name="HelloService">
  <types><!-- ... --></types>
  <message><!-- ... --></message>
  <portType><!-- ... --></portType>
  <binding><!-- ... --></binding>
  <service><!-- ... --></service>
</definitions>
```

WSDL `types`: Sample

The `types` element documents the XML Schema for the message parts.

Schema information can be nested or (more likely) imported.

```
<types>
    <xsd:schema>
        <xsd:import                                                    namespace="http://ou/"
   schemaLocation="http://localhost:7001/HelloWS/HelloService?xsd=1"/>
    </xsd:schema>
</types>
```

Make another HTTP `GET` request to obtain the XML Schema.

WSDL `types`: Sample Schema

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema xmlns:tns="http://ou/"
   xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0"
   targetNamespace="http://ou/">
    <xs:element name="getHello" type="tns:getHello"/>
    <xs:element name="getHelloResponse" type="tns:getHelloResponse"/>
    <xs:complexType name="getHello">
        <xs:sequence>
            <xs:element name="arg0" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="getHelloResponse">
        <xs:sequence>
            <xs:element name="return" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

This defines what the content of the SOAP body will look like.

WSDL `message`: Sample

The message elements list the parts that make up a message.

- Each part corresponds to an element in the XML Schema located in the `types` section.
- Older styles of SOAP messaging did not have a one-to-one mapping of messages to parts.

```
<message name="getHello">
    <part name="parameters" element="tns:getHello"/>
</message>
<message name="getHelloResponse">
    <part name="parameters" element="tns:getHelloResponse"/>
</message>
```

WSDL portType: Sample

Roughly, a portType corresponds to a Java class that is part of a web service.

```
<portType name="Hello">
    <operation name="getHello">
        <input
            wsam:Action="http://ou/Hello/getHelloRequest"
            message="tns:getHello"/>
        <output
            wsam:Action="http://ou/Hello/getHelloResponse"
            message="tns:getHelloResponse"/>
    </operation>
</portType>
```

An operation corresponds to a web method in the Java class.

Method parameters

Method return type

WSDL `binding`: Sample

The `binding` element controls how a SOAP request is structured. Using the wrong type means clients may not be able to construct messages for your service.

```
<binding name="HelloPortBinding" type="tns:Hello">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
                                                    style="document"/>
    <operation name="getHello">
        <soap:operation soapAction=""/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
    </operation>
</binding>
```

The SOAP body is a "document" instead of an rpc call.

Use XML data types instead of broken SOAP section 5 types.

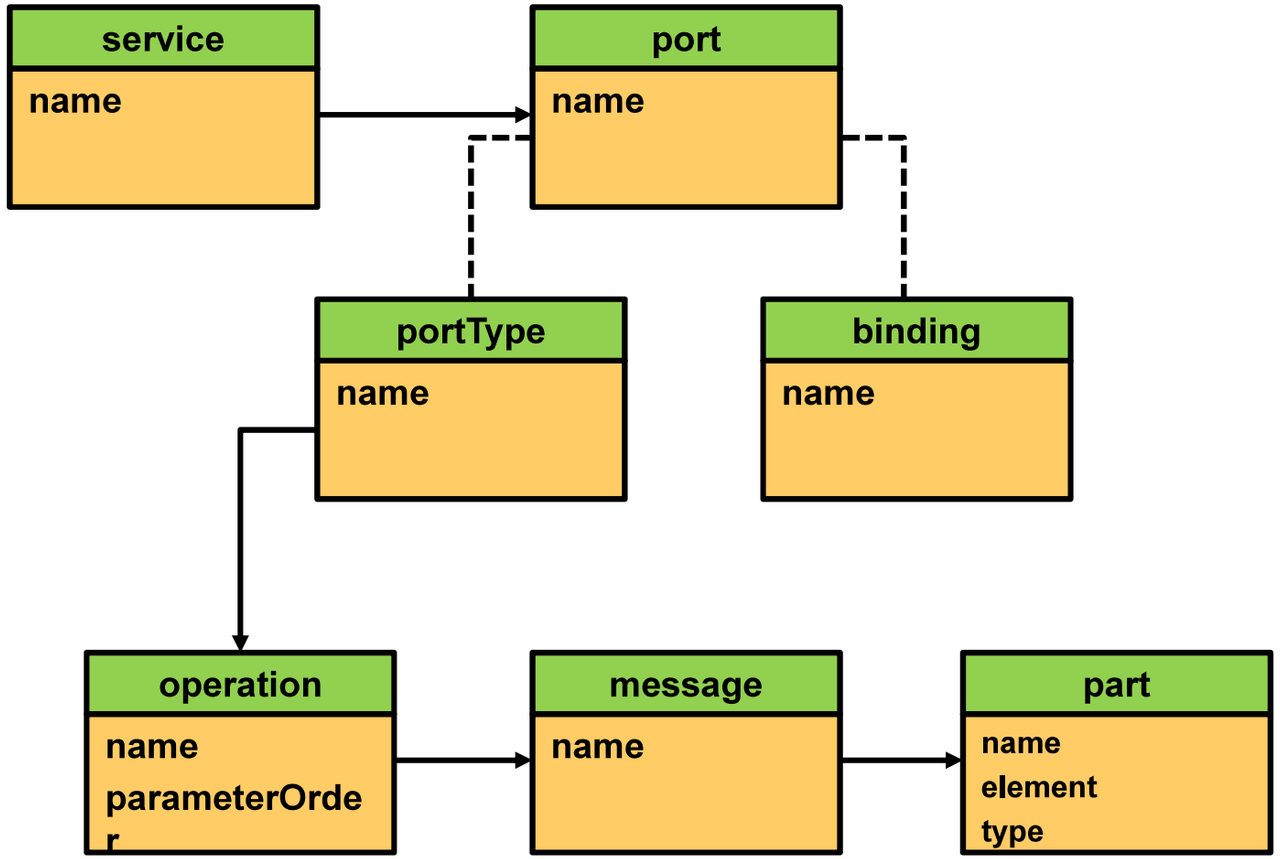
WSDL `service`: Sample

The `service` element controls what URLs your ports are available at. In practice, there is a one-to-one mapping of ports to services.

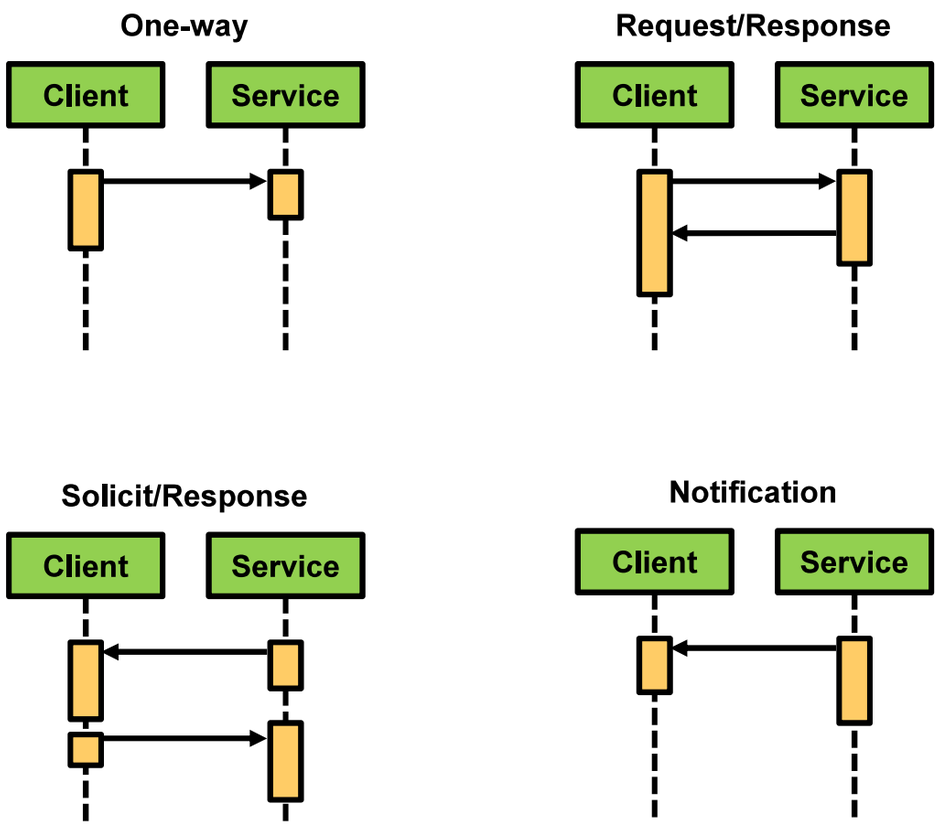
```
<service name="HelloService">
    <port name="HelloPort" binding="tns:HelloPortBinding">
        <soap:address
             location="http://localhost:7001/HelloWS/HelloService"/>
    </port>
</service>
```

Where SOAP over HTTP request are sent

Defining a Web Service in WSDL



WSDL Interaction Scenarios



Logical Versus Implementation Descriptions

- `<portType>`, `<operation>`, and `<message>` represent the *logical* description of a web service, that is, what the service can do.
- WSDL files also provide some implementation guidance:
  - XML Schemas define the representation of the data embedded in messages.
  - WSDL bindings provide additional guidance:
    - Style of WSDL to use

Variations of WSDL

Designers have to make two choices when writing WSDL descriptions for web services.

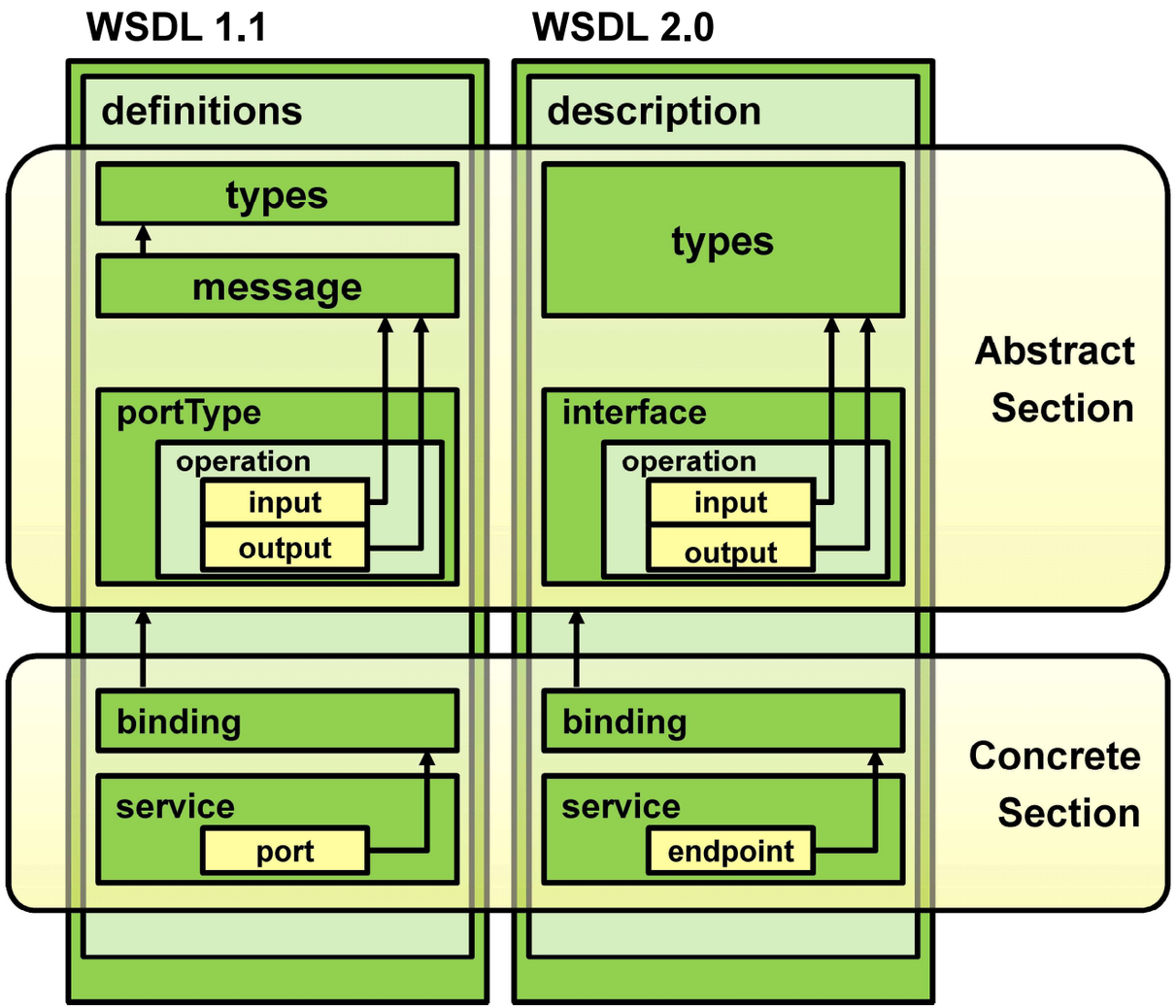
- What “style” of WSDL definition to use:
  - RPC-style
  - Document-style
- What data representation to use:
  - Literal
  - SOAP/RPC encoded (discouraged)

WS-I Basic Profile

The SOAP and WSDL specifications leave a lot of room for web services and clients to be able to construct incompatible messages, as well as different versions.

- Web Services Interoperability Organization (WS-I) is a group that attempts to define standards to promote interoperability between web service implementations.
- WS-I Basic Profile outlines a subset of features that should be used when developing web services.
  - For example, use document-literal-wrapped
- WebLogic 12*c* supports WS-I Basic Profile 2.0, 1.2, and 1.1.

WSDL 1.0 Versus WSDL 2.0



SOAP and HTTP

- SOAP can use other transport technologies.
  - SOAP over JMS http://www.w3.org/TR/soapjms/
- Most realizations of SOAP use HTTP.
- HTTP POST:
  - SOAP message is the request/response body.
  - SOAPAction HTTP header helps identify the operation.
    - No longer required in SOAP 1.2.
- Any tool that can be used to send a POST request can function as a SOAP web service testing tool.
  - The SOAP XML elements are small and rarely change.
  - The SOAP body content is just XML that is constrained by the XML Schemas embedded in the WSDL `types` section.

SOAP Test Clients

WebLogic Service includes a web-based web service test client application: http://localhost:7001/wls_utc/.

WS-* Extensions

WS-I Basic Profile outlines a basic set of interoperable functionality. Many add-on standards have been created:

- Web Services Policy Framework (WS-Policy)
- Web Services Policy Attachment (WS-PolicyAttachment)
- Web Services Security (WS-Security)
- Web Services Security Policy (WS-SecurityPolicy)
- Web Services Addressing (WS-Addressing)
- Web Services Reliable Messaging (WS-ReliableMessaging)
- Web Services Reliable Messaging Policy Assertion (WS-RM Policy)
- Web Services Secure Conversation Language (WS-SecureConversation)
- WS-MakeConnection
- Web Services Atomic Transaction
- Message Transmission Optimization Mechanism (MTOM)

WS-Policy

A WSDL outlines the basic contract a client must follow when communicating with a web service.

- If additional restrictions or requirements for clients are in place then, ideally, they would be included in the WSDL.
- WS-Policy is a specification that outlines how extra constraints can be specified by using XML added to a WSDL.
- WS-Policy is open ended. It specifies how WS-* extensions can add restrictions.
- WS-Policy and WS-* enhancements are not covered by JAX-WS. If and how they work are vendor specific.
- WebLogic provides a `@weblogic.jws.Policy` annotation

@weblogic.jws.Policy

Used to attach WS-Policy requirements by developers.

```
@Policy(uri="policy:Mc1.1.xml", attachToWsdl=true)
```

- Developers can choose to add the policy element to the generated WSDL or not.
- WebLogic Service includes a large number of prewritten policies that should be preferred over custom policies.
- Administrators can attach policies to deployed web services.

http://docs.oracle.com/cd/E24329_01/apirefs.1211/e24401/taskhelp/webservices/ConfigureWSPolicyFile.html

Quiz

A request to a SOAP web service endpoint must always be transferred using the HTTP protocol.

a. True

b. False

Quiz

The definitions element in a completed WSDL should contain which sequence of elements?

a. types, message, portType, binding, service
b. Envelope, Header, Body
c. sequence, choice, enumeration, restriction
d. types, parts, ports, binding, location

Resources

Topic	Website
Simple Object Access Protocol (SOAP) 1.1	http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
Web Services Description Language (WSDL) Version 1.2	http://www.w3.org/TR/2002/WD-wsdl12-20020709/
Web Services Policy 1.5 - Framework	http://www.w3.org/TR/ws-policy/

Summary

In this lesson, you should have learned how to:

- Describe the basic structure of a Simple Object Access Protocol (SOAP) message and how it is encapsulated by transports
- Explain how WSDL defines a web service, including its message representation and transport mechanism
- Explain the purpose of WS-I Basic Profile and WS-Policy

Practice 4: Overview

This practice covers the following topics:

- Revisiting the Calculator Web Service
- Configuring WebLogic for WS-* Web Services
- Exploring SOAP and WSDL Documents with WS-* Extensions