Cloud Computing: Introduction to Web Services

Vijay Dialani, PhD
Boise State University
vijaydialani@boisestate.edu

Cloud Computing – git repository location

- https://github.com/vdialani/CloudComputing.git
- Look for examples in “chapter1” folder of this repository.

Cloud Computing – Introduction to Web Services

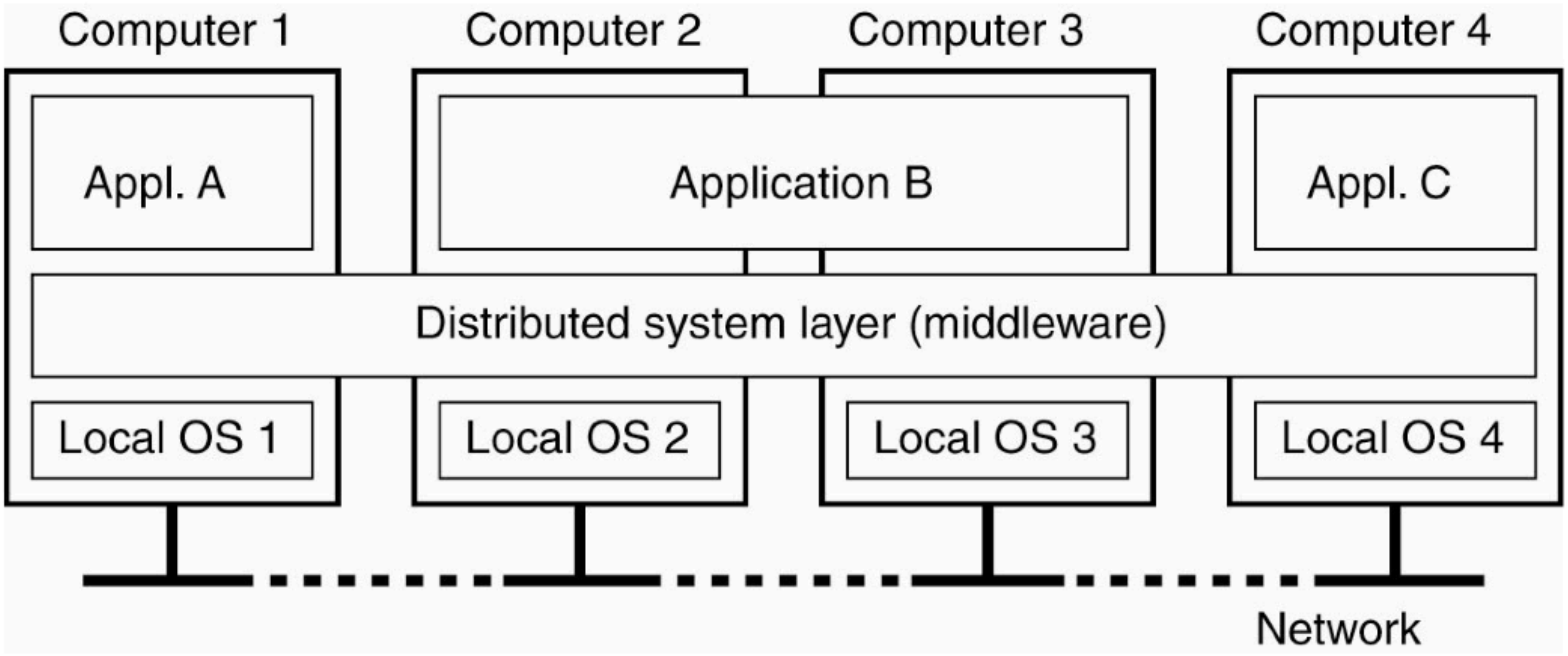
What is a Web Service?

- Web Service is a communication protocol that allows components in distributed systems to exchange messages in accordance to its published interface (specifically WSDL).

Recap: Distributed Systems I

- A Distributed System consists of multiple computational components that communicate over a network in order to achieve a common objective.
- Communication between the computers is accomplished by exchange of messages. The semantics and syntax of these messages is specified using a common interface description language.
- Components in a distributed system operate concurrently, do not have to be collocated or in same time zone (clocks need not be synchronized) and nodes may fail and recover independently of each other.
- There may be no node in the system with complete knowledge of all the computational tasks or the data being handled by the system.
- A distributed system may suffer network partitioning and needs to degrade its performance gracefully in face of such failures.

Recap: Distributed Systems II



- Distributed System rely on middleware to provide a coherent framework of services to the applications.

Recap: Distributed Systems III

A Distributed System Middleware:

- Provides a Resource Discovery Service
- Allows transparent resource migration
- Provides the secure context for execution by identifying and authenticating the users and systems
- Provides redundancy and replication to hide movement of resources while the system is in use.

Implementations of distributed middleware

- CORBA (Common Object Resource Broker Architecture)
- DCOM (Distributed Common Object Model)
- JAVA-RMI (Java Remote Method Invocation)
- Message Queuing Systems (Rabbit MQ, MSMQ, IBM MQ Series)

Recap: Distributed System IV

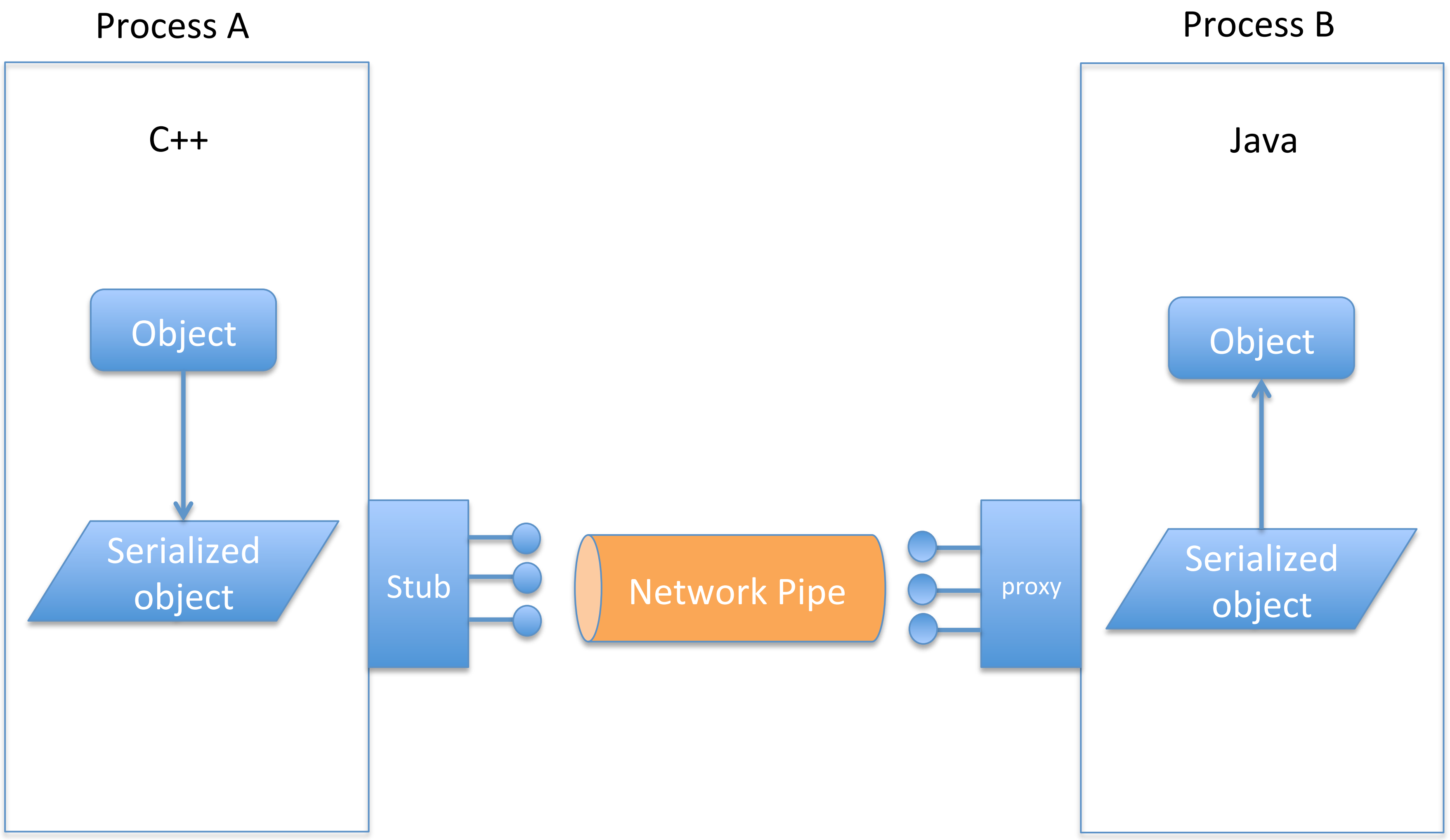
- Drawbacks of previous approaches
    - Vendor specific solutions lack of interoperability.
    - Different binary formats meant many adapter needed to be implemented.
    - Hard to debug, efficiency requirements dominated and all the data was exchanged in binary format

Recap: Distributed Systems V

Serialization

- A way to automatically save and recreate the state of the object.
- Serialization converts an object into a byte buffer that could be sent to other systems or could be saved to disc for recreation of the object.
- De-serialization converts the byte buffer to an object instance.
- Serialization can maintain either a shallow copy or a deep copy of the object.
- In a deep copy the nested objects are also serialized and de-serialized to save or recreate an object
- Each class used for serialization has an associated serialization ID to verify that the serialized and deserialization process are loading the same version of the class

Recap: Remote Procedure Call



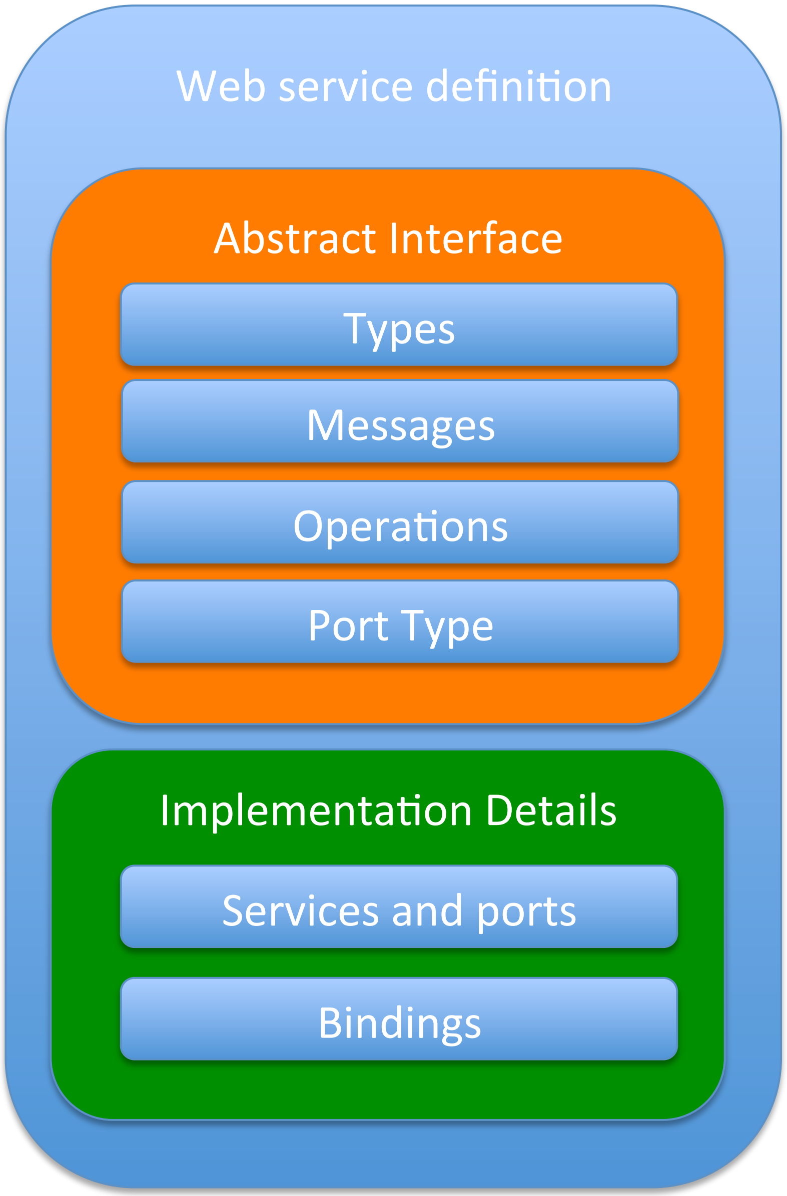
Web Services Description Language

A web service description contains **definitions** about:

- simple and complex **data types** supported by the service
- **messages** that will be exchanged between the server and the client and will be expressed in terms of data types defined and imported by the service
- interfaces i.e. **port types** that will be used to exchange these messages between the client and the server program
- **bindings** that specify the data format and protocol for each of the port types

A WSDL specification is an XML document

Web Service Definition



Abstract Interface describes the type system, the messages, the composition of these messages into operations and ports.

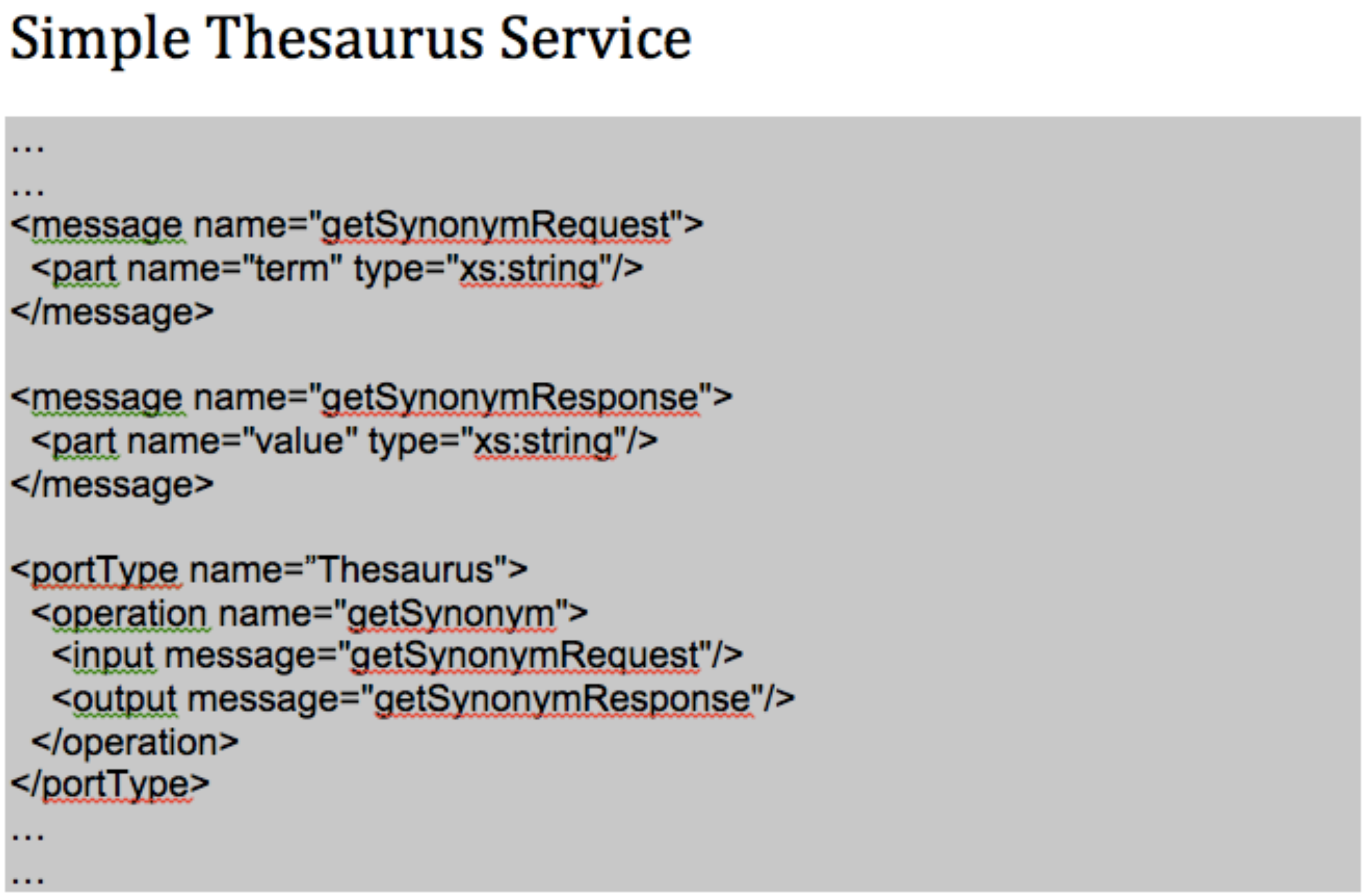
It imposes no requirement of how these messages are exchanged.

Different service implementations may support the same interface

Implementation Details specify the wire format that will be used by the web service instance. The location or endpoint details of the service instance.

It is purely service instance specific information.

Web Services : Simple Thesaurus Service



Simple Thesaurus Service

```
…
…
<message name="getSynonymRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getSynonymResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="Thesaurus">
  <operation name="getSynonym">
    <input message="getSynonymRequest"/>
    <output message="getSynonymResponse"/>
  </operation>
</portType>
…
…
```

Web Services: Type System

Simple Data Structure

```
struct customerInfo {
        string name;
        int age;
        string phone;
        enum shipping { “regular”, “2-day”, “1-day” };
};
```

Imposing format restrictions

```
<simpleType name="phone">
        <restriction base="xsd:string">
                <pattern value="\d{3}-\d{3}-\d{4}"/>
        </restriction>
</simpleType>
```

Web Services: Type System

Declaring Enumerations

```
<simpleType name="shipping">
        <restriction base="xsd:string">
                <enumeration value="regular"/>
                <enumeration value="2-day"/>
                <enumeration value="1-day"/>
        </restriction>
</simpleType>
```

Declaring Complex Derived Type

```
<complexType name="customerInfo">
        <sequence>
        <element name="name" type="xsd:string" />
        <element name="age" type="xsd:int" />
        <element name="phone_number" type="phone" />
        <element name="shipping_method" type="shipping" />
        </sequence>
</complexType>
```

Web Services: Type System

Allowing only one by using choice

```
<complexType name="customerInfo">
        <choice>
                <element name="name" type="xsd:string"/>
                <element name="age" type="xsd:int"/>
                <element name="phone_number" type="phone" />
                <element name="shipping_method" type="shipping"/>
        </choice>
</complexType>
```

Imposing array length restrictions

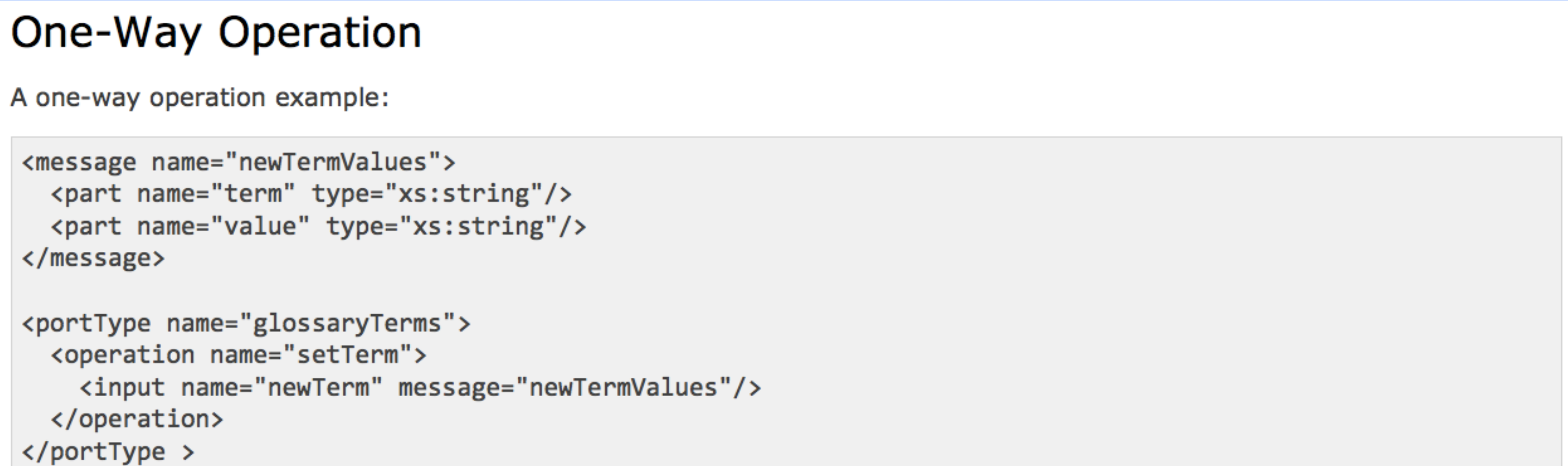
```
<complexType name="customerInfo">
        <all>
                <element name="name" type="xsd:string"/>
                <element name="age" type="xsd:int"/>
                <element name="phone_number" type="phone" minOccurs="3"
maxOccurs="7"/>
                <element name="shipping_method" type="shipping"/>
        </all>
</complexType>
```

Web Services Definition

Operation Types:

Type	Definition
One Way	The operation can receive a message but will not return a response
Request Response	The operation can receive a request and will return a response
Solicit Response	The operation can send a request and will wait for a response
Notification	The operation can send a message but will not wait for a response

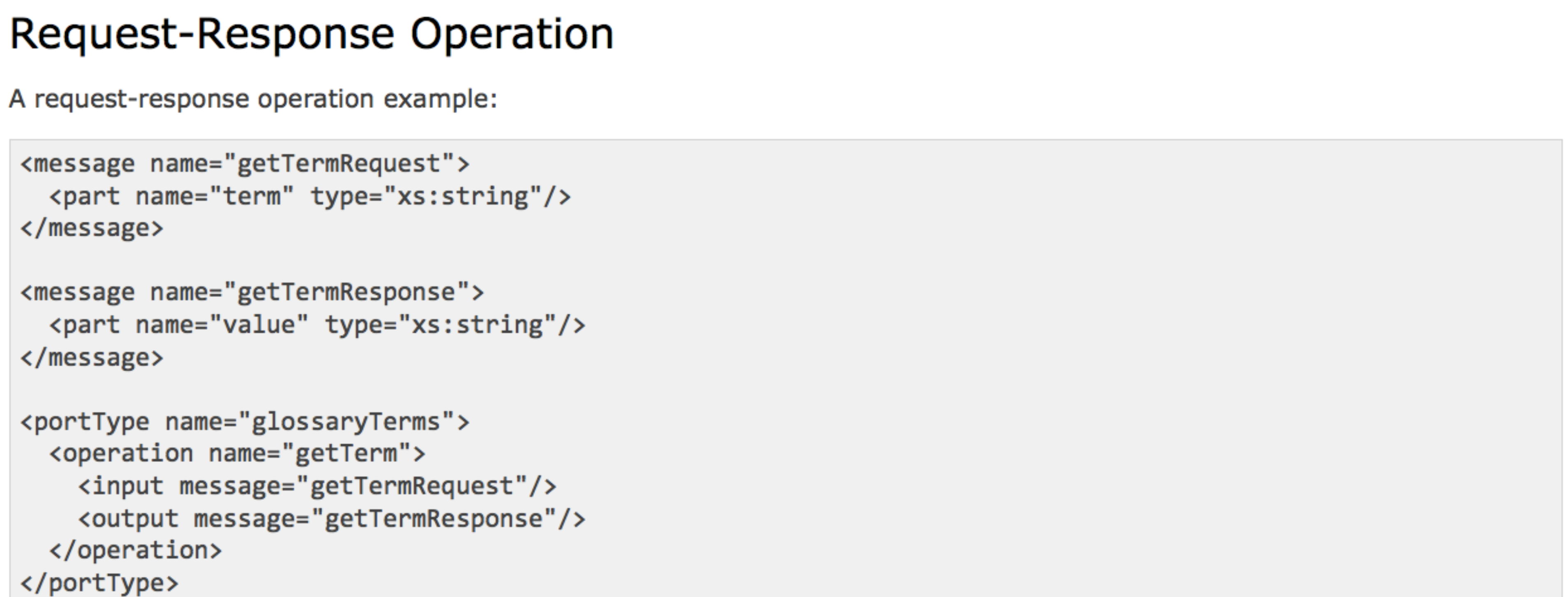
Web Services Operation Types



One-Way Operation

A one-way operation example:

```
<message name="newTermValues">
  <part name="term" type="xs:string"/>
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="setTerm">
    <input name="newTerm" message="newTermValues"/>
  </operation>
</portType >
```

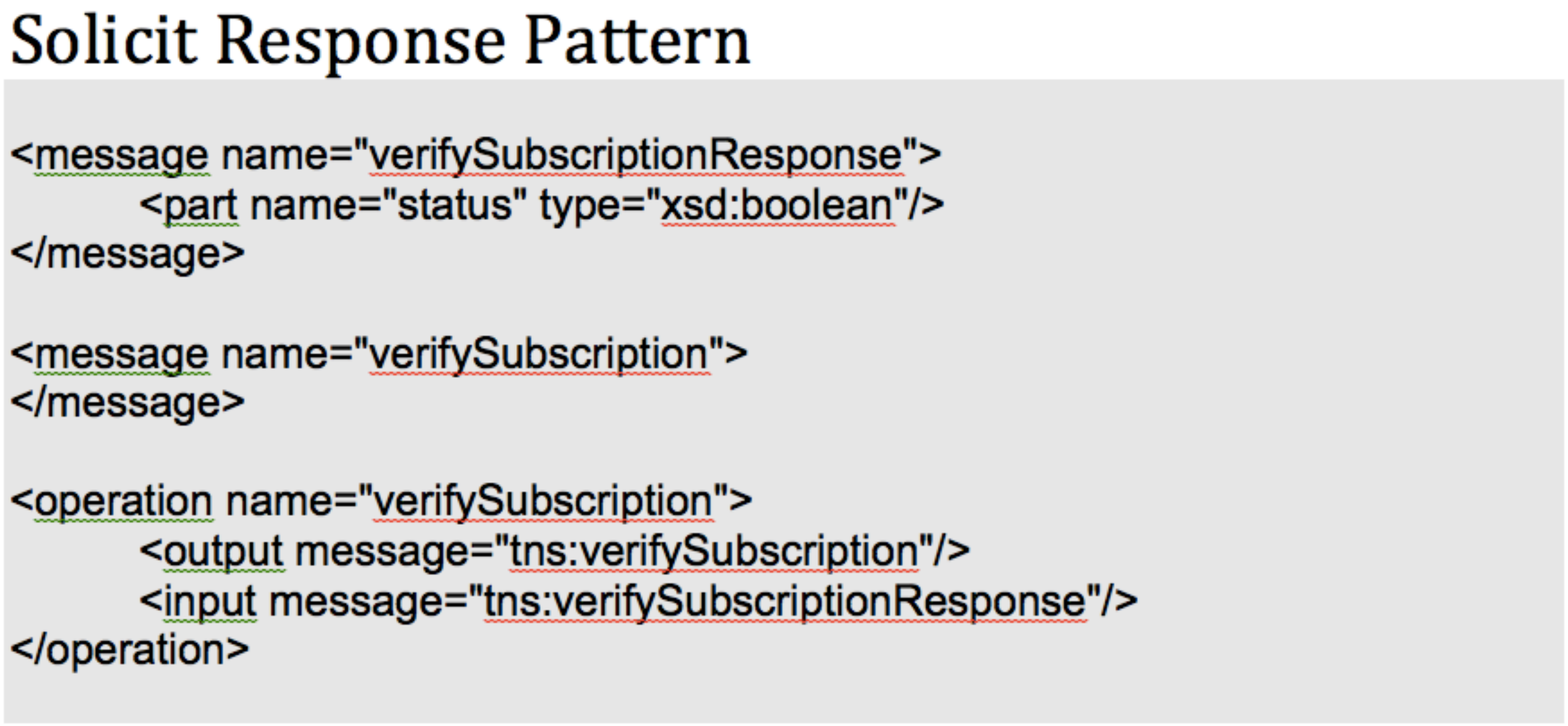


Request-Response Operation

A request-response operation example:

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

Web Services Operation Types



Solicit Response Pattern

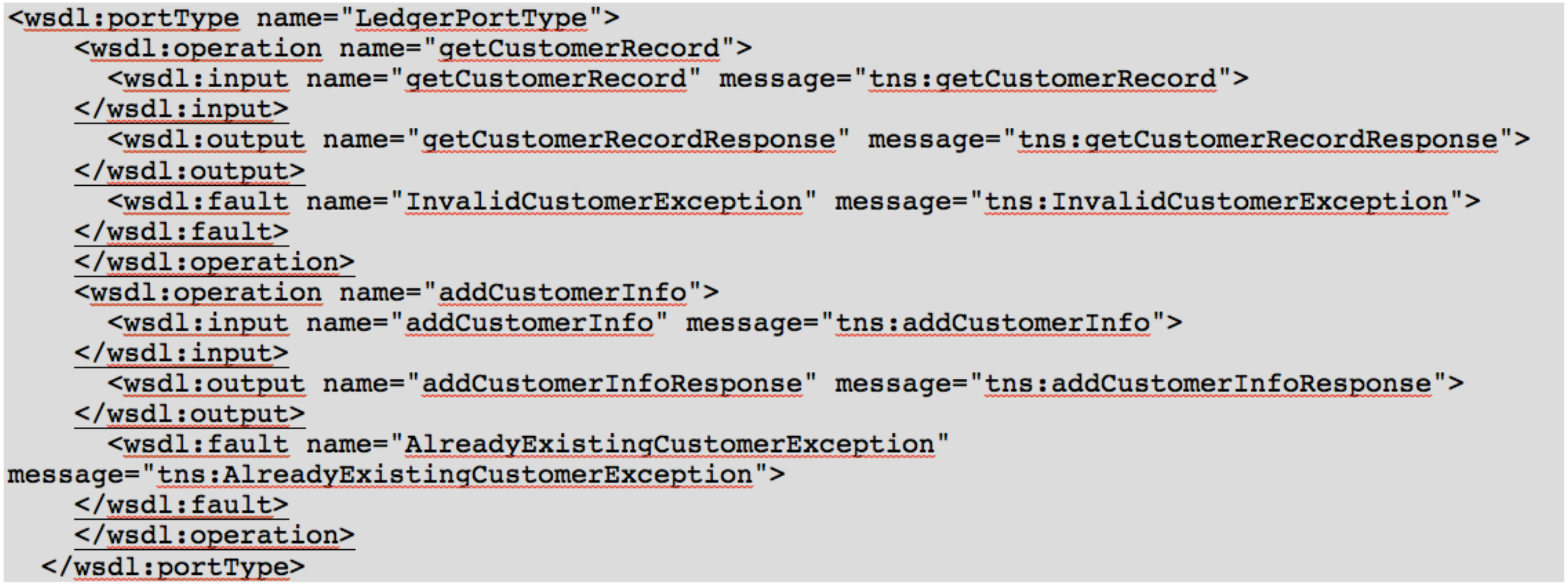
```
<message name="verifySubscriptionResponse">
        <part name="status" type="xsd:boolean"/>
</message>

<message name="verifySubscription">
</message>

<operation name="verifySubscription">
        <output message="tns:verifySubscription"/>
        <input message="tns:verifySubscriptionResponse"/>
</operation>
```

Web Services Ports

```
package course.cloud.computing;

public interface Ledger {

// This method gets information of customer with the specified
id, if the customer is not found an exception is raised
      public CustomerInfo getCustomerRecord(int id) throws
      InvalidCustomerException;

// This method adds information of customer and returns a newly
allocated id, if the customer already exists an exception is
raised
      public int addCustomerInfo(CustomerInfo info) throws
      AlreadyExistingCustomerException;
}
```

Web Services

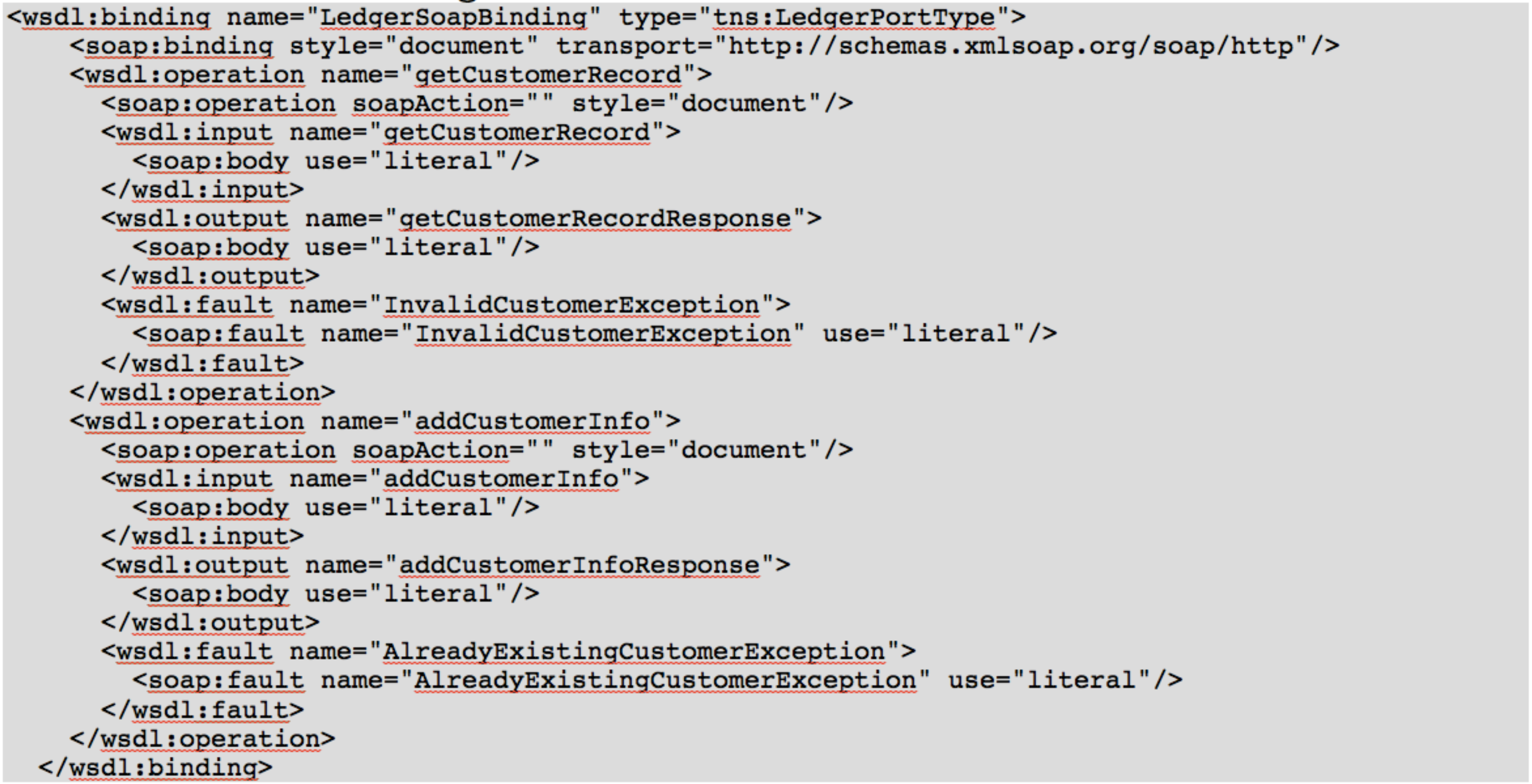
Port Type Definition



```
<wsdl:portType name="LedgerPortType">
    <wsdl:operation name="getCustomerRecord">
      <wsdl:input name="getCustomerRecord" message="tns:getCustomerRecord">
    </wsdl:input>
      <wsdl:output name="getCustomerRecordResponse" message="tns:getCustomerRecordResponse">
    </wsdl:output>
      <wsdl:fault name="InvalidCustomerException" message="tns:InvalidCustomerException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="addCustomerInfo">
      <wsdl:input name="addCustomerInfo" message="tns:addCustomerInfo">
    </wsdl:input>
      <wsdl:output name="addCustomerInfoResponse" message="tns:addCustomerInfoResponse">
    </wsdl:output>
      <wsdl:fault name="AlreadyExistingCustomerException"
message="tns:AlreadyExistingCustomerException">
    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
```

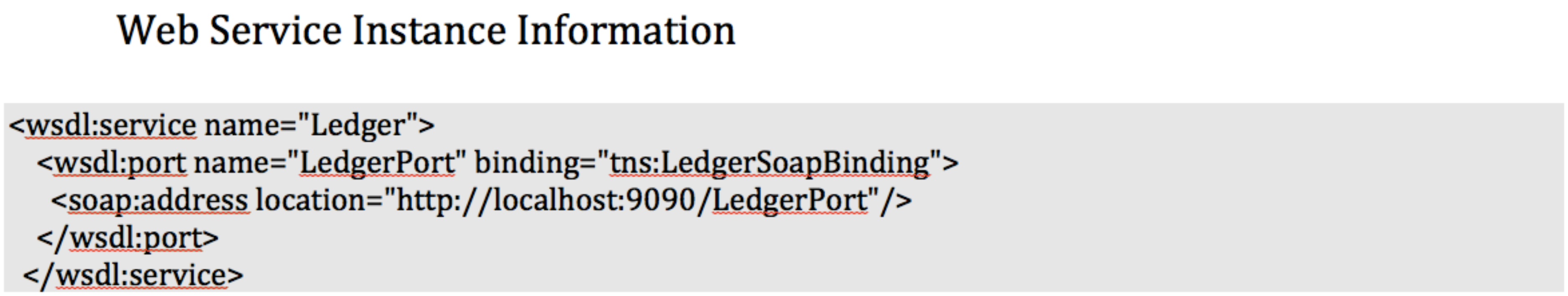
Web Service Binding

WSDL SOAP Binding



```
<wsdl:binding name="LedgerSoapBinding" type="tns:LedgerPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getCustomerRecord">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="getCustomerRecord">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getCustomerRecordResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="InvalidCustomerException">
        <soap:fault name="InvalidCustomerException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="addCustomerInfo">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="addCustomerInfo">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="addCustomerInfoResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="AlreadyExistingCustomerException">
        <soap:fault name="AlreadyExistingCustomerException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
```

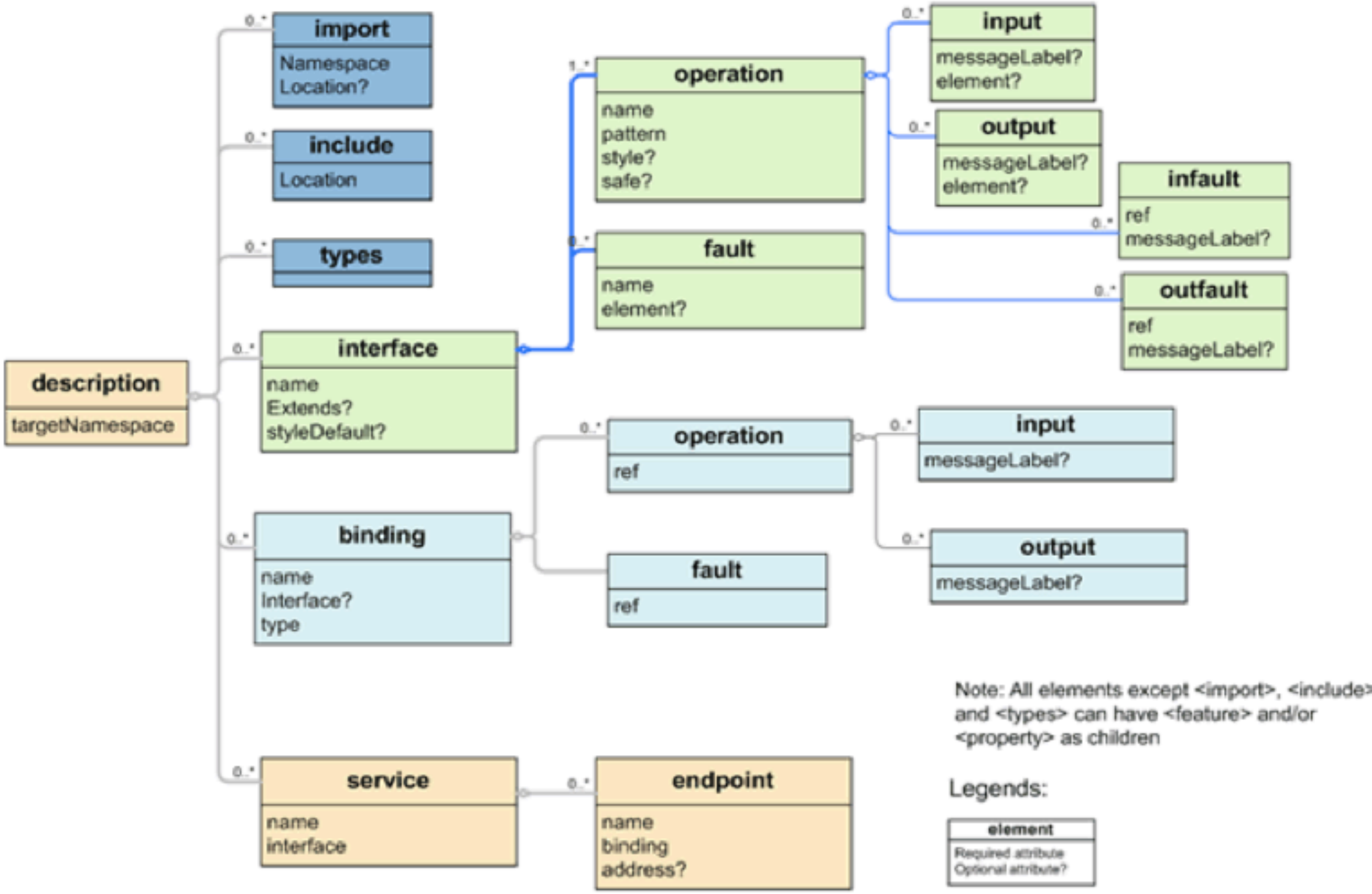
Web Services - Service



Web Service Instance Information

```
<wsdl:service name="Ledger">
  <wsdl:port name="LedgerPort" binding="tns:LedgerSoapBinding">
   <soap:address location="http://localhost:9090/LedgerPort"/>
  </wsdl:port>
 </wsdl:service>
```

Web Services – XML Namespace Representation



Note: All elements except <import>, <include
and <types> can have <feature> and/or
<property> as children

Legends:

element
Required attribute Optional attribute?

Web Services

WebServices Description Language (WSDL) Version 2.0 Part1:CoreLanguage:
http://www.w3.org/TR/wsdl20/#intro

First Look at WSDL2.0:
https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/74bae690-0201-0010-71a5-9da49f4a53e2