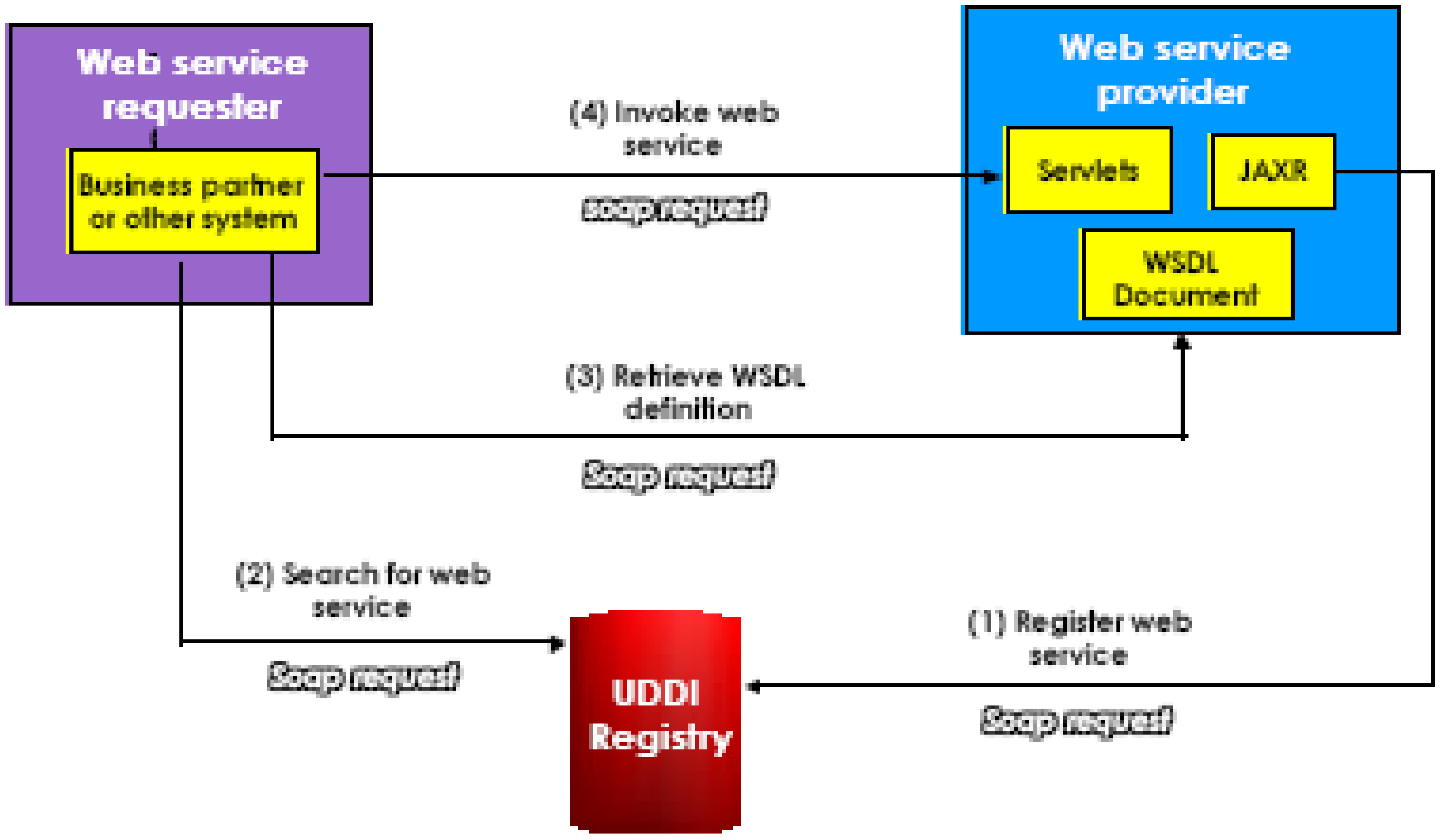
WSDL

Java Perspective

What is WSDL?

- XML language for describing web services
- Web service is described as
  - A set of communication endpoints (ports)
- Endpoint is made of two parts
  - Abstract definitions of operations and messages
  - Concrete binding to networking protocol (and corresponding endpoint address) and message encoding
- Why this separation?
  - Enhance reusability (as we will see in UDDI reference to WSDL document)

Where is WSDL Used



Why WSDL?

- Enables automation of communication details between communicating partners
  - Machines can read WSDL
  - Machines can invoke a service defined in WSDL
- Discoverable through registry
- Arbitration
  - 3rd party can verify if communication conforms to WSDL

WSDL Document Structure

WSDL Document Structure

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl"
targetNamespace="your namespace here"
xmlns:tns="your namespace here"
xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap">
    <wsdl:types>
    <xs:schema targetNamespace="your namespace here (could be another) "
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    <!-- Define types and possibly elements here -->
    </schema>
    </wsdl:types>
            <wsdl:message name="some operation input">
            <!-- part(s) here -->
            </wsdl:message>
            <wsdl:message name="some operation output">
            <!-- part(s) here -->
            </wsdl:message>
    <wsdl:portType name="your type name">
            <!-- define operations here in terms of their messages -->
    </wsdl:portType>
    <wsdl:binding name="your binding name" type="tns:port type name above">
            <!-- define style and transport in general and use per operation -->
    </wsdl:binding>
    <wsdl:service>
            <!-- define a port using the above binding and a URL -->
    </wsdl:service>
    </wsdl:definitions>
```

WSDL Namespaces

- http://schemas.xmlsoap.org/wsdl
- http://schemas.xmlsoap.org/wsdl/soap
- http://www.w3.org/2001/XMLSchema

WSDL Document Example

- Simple service providing stock quotes
- A single operation called *GetLastTradePrice*
- Deployed using SOAP 1.1 over HTTP
- Request takes a ticker symbol of type string
- Response returns price as a float

WSDL Elements

- Types
- Message
- Port Type
  - Operation
- Binding
- Port
- Service

WSDL Elements

- Types
  - Data type definitions
  - Used to describe exchanged messages
  - Uses W3C XML Schema as canonical type system

WSDL Example: Types

```
<definitions name="StockQuote"
targetNamespace="http://example.com/stockquote.wsdl"
xmlns:tns="http://example.com/stockquote.wsdl"
xmlns:xsd1="http://example.com/stockquote.xsd"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>
	<schema targetNamespace="http://example.com/stockquote.xsd"
	xmlns="http://www.w3.org/2000/10/XMLSchema">
			<element name="TradePriceRequest">
					<complexType>
							<all>
							<element name= "tickerSymbol" type="string"/>
							</all>
					</complexType>
			</element>
			<element name="TradePrice">
					<complexType>
							<all>
							<element name="price" type="float"/>
							</all>
					</complexType>
			</element>
</schema>
</types>
```

WSDL Elements

- Messages
  - Abstract, typed definitions of data being exchanged
- Operations
  - Abstract description of an action
  - Refers to an input and/or output messages
- Port type
  - Collection of operations
  - Abstract definition of a service

Example: Messages, Operation, Port type

```
<message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>
    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>
    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
                <input message="tns:GetLastTradePriceInput"/>
                <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>
```

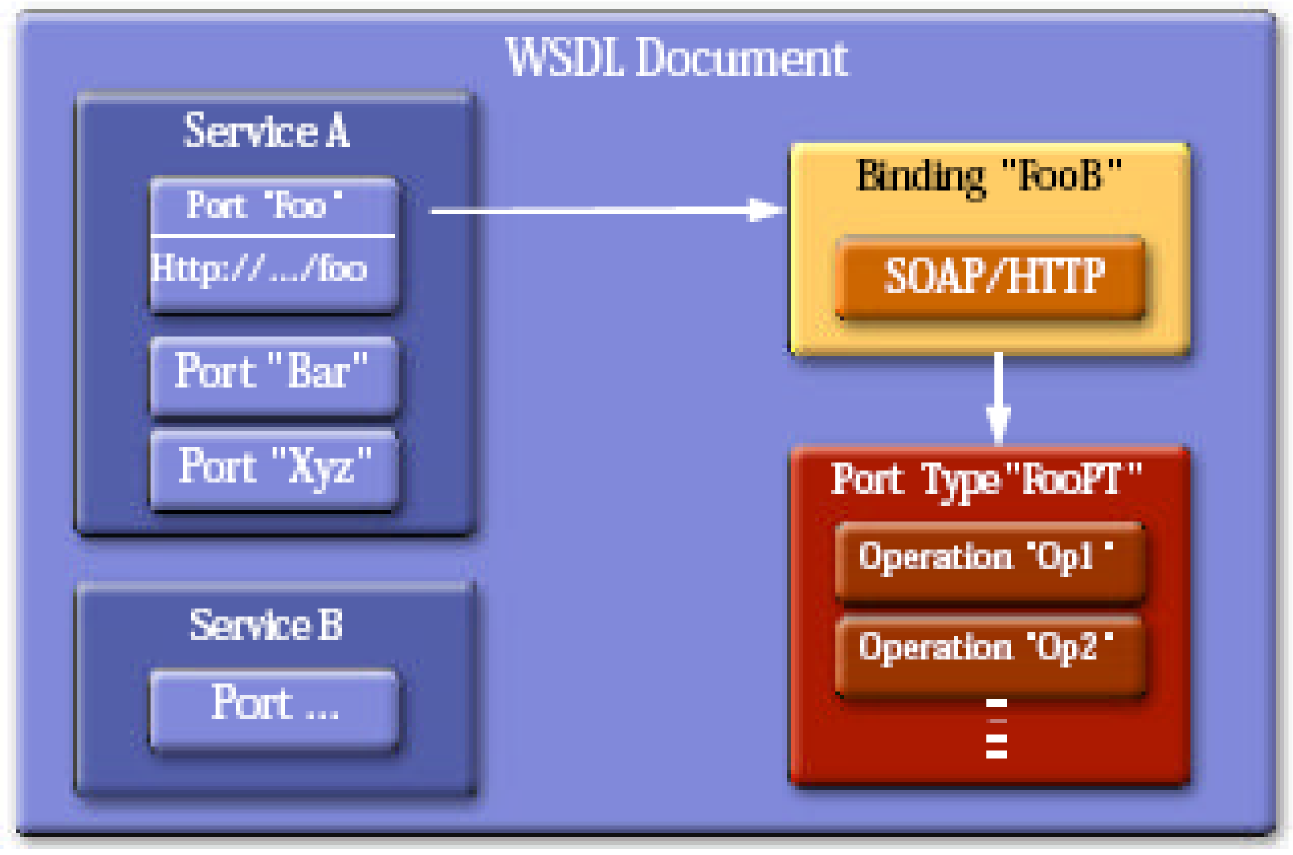
WSDL Elements

- Binding
  - Concrete protocol and data format (encoding) for a particular Port type
- Protocol examples: SOAP 1.1 over HTTP or SOAP 1.1 over SMTP
- Encoding examples: SOAP encoding, RDF encoding
- Port
  - Defines a single communication endpoint
  - Endpoint address for binding
  - URL for HTTP, email address for SMTP
- Service
  - Aggregate set of related ports

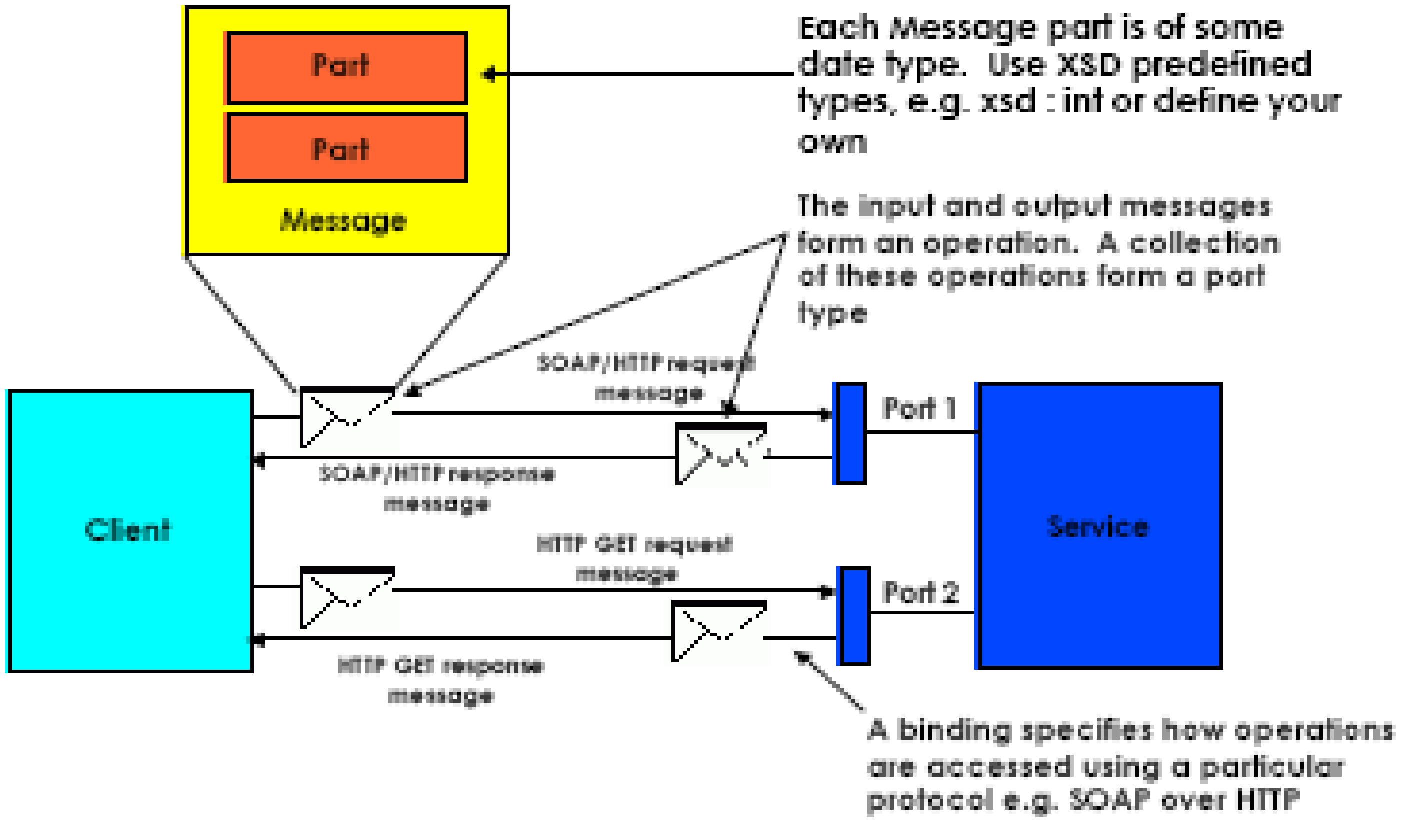
Example: Binding, Port, Service

```
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
		<soap:binding style="document"
	transport="http://schemas.xmlsoap.org/soap/http"/>
		<operation name="GetLastTradePrice">
			<soap:operation
	soapAction="http://example.com/GetLastTradePrice"/>
			<input>
				<soap:body use="literal"/>
			</input>
			<output>
				<soap:body use="literal"/>
			</output>
		</operation>
	</binding>
	<service name="StockQuoteService">
		<documentation>My first service</documentation>
		<port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
			<soap:address location="http://example.com/stockquote"/>
		</port>
	</service>
```

WSDL View of a Web Service



Web Service Invocation


Part
Part
Message
Each Message part is of some date type. Use XSD predefined types, e.g. xsd : int or define your own
The input and output messages form an operation. A collection of these operations form a port type
SOAP/HTTP request message
SOAP/HTTP response message
Port 1
Client
Service
HTTP GET request message
Port 2
HTTP GET response message
A binding specifies how operations are accessed using a particular protocol e.g. SOAP over HTTP

Message Element

- Consist of one or more logical parts
- Syntax

```
<definitions .... >
<message name="nmtoken"> *
<part name="nmtoken" element="qname"?
   type="qname"?/> *
</message>
</definitions>
```

  - *element* attribute refers to an XSD element using a QName
  - *type* attribute refers to an XSD *simpleType* or *complexType* using a QName

Message Element

```
<definitions>
        <types>
        <schema>
        <element name="PO" type="tns:POType"/>
                <complexType name="POType">
                <all>
                <element name="id" type="string"/>
                <element name="name" type="string"/>
                <element name="items">
                <complexType>
                <all>
                <element name="item" type="tns:Item" minOccurs="0" maxOccurs="unbounded"/>
                </all>
                </complexType>
                </element>
                </all>
                </complexType>
                <complexType name="Item">
                <all>
                <element name="quantity" type="int"/>
                <element name="product" type="string"/>
                </all>
                </complexType>
        <element name="Invoice" type="tns:InvoiceType"/>
                <complexType name="InvoiceType">
                <all>
                <element name="id" type="string"/>
                </all>
                </complexType>
                </schema>
        </types>
        <message name="PO">
                <part name="po" element="tns:PO"/>
                <part name="invoice" element="tns:Invoice"/>
        </message>
</definitions>
```

Types of Operations

- One-way
  - The endpoint receives a message
- Request/response
  - The endpoint receives a message, and sends a correlated message
- Notification
  - The endpoint sends a message
- Solicit/response
  - The endpoint sends a message, and receives a correlated message

One-way Operation

```
<operation name=”submitPurchase”>
<input message=”purchase”/>
</operation>
```

Request/Response Operation

```
<operation name=”submitPurchase”>
    <input message=”purchase”/>
    <output message=”confirmation”/>
</operation>
<operation name=”submitPurchase”>
    <input message=”purchase”/>
    <output message=”confirmation”/>
    <fault message=”faultMessage”/>
</operation>
```

Notification Operation

```
<operation name=”deliveryStatus”>
<output message=”trackingInformation”/>
</operation>
```

Solicit/Response Operation

```
<operation name=”clientQuery”>
<output message=”bandwidthRequest”/>
<input message=”bandwidthInfo”/>
<fault message=”faultMessage”/>
</operation>
```

Binding Element

Binding Element

- Defines protocol details and message format for *operations* and *messages* defined by a particular *portType*
- Specify one protocol out of
  - SOAP (SOAP over HTTP, SOAP over SMTP)
  - HTTP GET/POST
- Provides extensibility mechanism
  - Can includes binding extensibility elements
  - Binding extensibility elements are used to specify the concrete grammar

Binding Element Syntax

```
<wsdl:definitions ..>
<wsdl:binding name="nmtoken" type="qname"> *
            <-- extensibility element per binding --> *
<wsdl:operation name="nmtoken"> *
            <-- extensibility element per operation --> *
<wsdl:input name="nmtoken"? > ?
            <-- extensibility element per input -->
</wsdl:input>
<wsdl:output name="nmtoken"? > ?
            <-- extensibility element per output --> *
</wsdl:output>
<wsdl:fault name="nmtoken"> *
            <-- extensibility element per fault --> *
</wsdl:fault>
</wsdl:operation>
</wsdl:binding>
</wsdl:definitions>
```

SOAP Binding

SOAP Binding Extension

- WSDL includes binding for SOAP 1.1 endpoints and supports:
  - Indication of binding to SOAP as a protocol
  - Address for SOAP endpoint
  - The URI for SOAPAction HTTP header (applies only for HTTP binding of SOAP)
  - List of definitions for Headers for SOAP envelope
- “soap” namespace
  - xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"

SOAP Binding Extensions Syntax

```
<definitions .... >
<binding .... >
<soap:binding style="rpc|document" transport="uri">
<operation .... >
<soap:operation soapAction="uri"? style="rpc|document"?>?
<input>
<soap:body parts="nmtokens"? use="literal|encoded" encodingStyle="uri-list"? namespace="uri"?>
<soap:header message="qname" part="nmtoken" use="literal|encoded"
encodingStyle="uri-list"? namespace="uri"?>*
<soap:headerfault message="qname" part="nmtoken" use="literal|encoded"
encodingStyle="uri-list"? namespace="uri"?/>*
<soap:header>
</input>
<output>
<soap:body parts="nmtokens"? use="literal|encoded" encodingStyle="uri-list"? namespace="uri"?>
<soap:header message="qname" part="nmtoken" use="literal|encoded"
encodingStyle="uri-list"? namespace="uri"?>*
<soap:headerfault message="qname" part="nmtoken" use="literal|encoded"
encodingStyle="uri-list"? namespace="uri"?/>*
<soap:header>
</output>
<fault>*
<soap:fault name="nmtoken" use="literal|encoded" encodingStyle="uri-list"? namespace="uri"?>
</fault>
</operation>
</binding>
<port .... >
<soap:address location="uri"/>
</port>
</definitions>
```

soap:binding

```
<definitions .... >
<binding .... >
<soap:binding transport="uri"? style="rpc|document"?>
</binding>
</definitions>
```

- Must be present when using SOAP binding
- *style* attribute applies to each contained operation (default: *document*) unless it is overidden by operation specific *style* attribute
- *transport* attribute indicates which transport to use
  - http://schemas.xmlsoap.org/soap/http (for HTTP)
  - http://schemas.xmlsoap.org/soap/smtp (for SMTP)

soap:operation

```
<definitions .... >
    <binding .... >
    <operation .... >
        <soap:operation soapAction="uri"? style="rpc|
    document"?>?
    </operation>
    </binding>
</definitions>
```

- *style* attribute indicates whether the operation is RPCoriented (messages containing parameters and return values) or document-oriented (message containing document(s))
  - Affects the way in which the Body of the SOAP message is constructed on the wire
- *soapAction* attribute specifies the value of the *SOAPAction* header for this operation

soap:body

```
<definitions .... >
<binding .... >
<operation .... >
<input>
<soap:body parts="nmtokens"? use="literal|encoded"?
encodingStyle="uri-list"? namespace="uri"?>
</input>
<output>
<soap:body parts="nmtokens"? use="literal|encoded"?
encodingStyle="uri-list"? namespace="uri"?>
</output>
</operation>
</binding>
</definitions>
```

soap:body

- Specifies how the message parts appear inside the SOAP Body element
  - Provides information on how to assemble the different message parts inside the Body element
- Used in both RPC-oriented and document-oriented messages
  - Which one to use is determined via *style* attribute of soap:binding or soap:operation elements

soap:body for RPC style

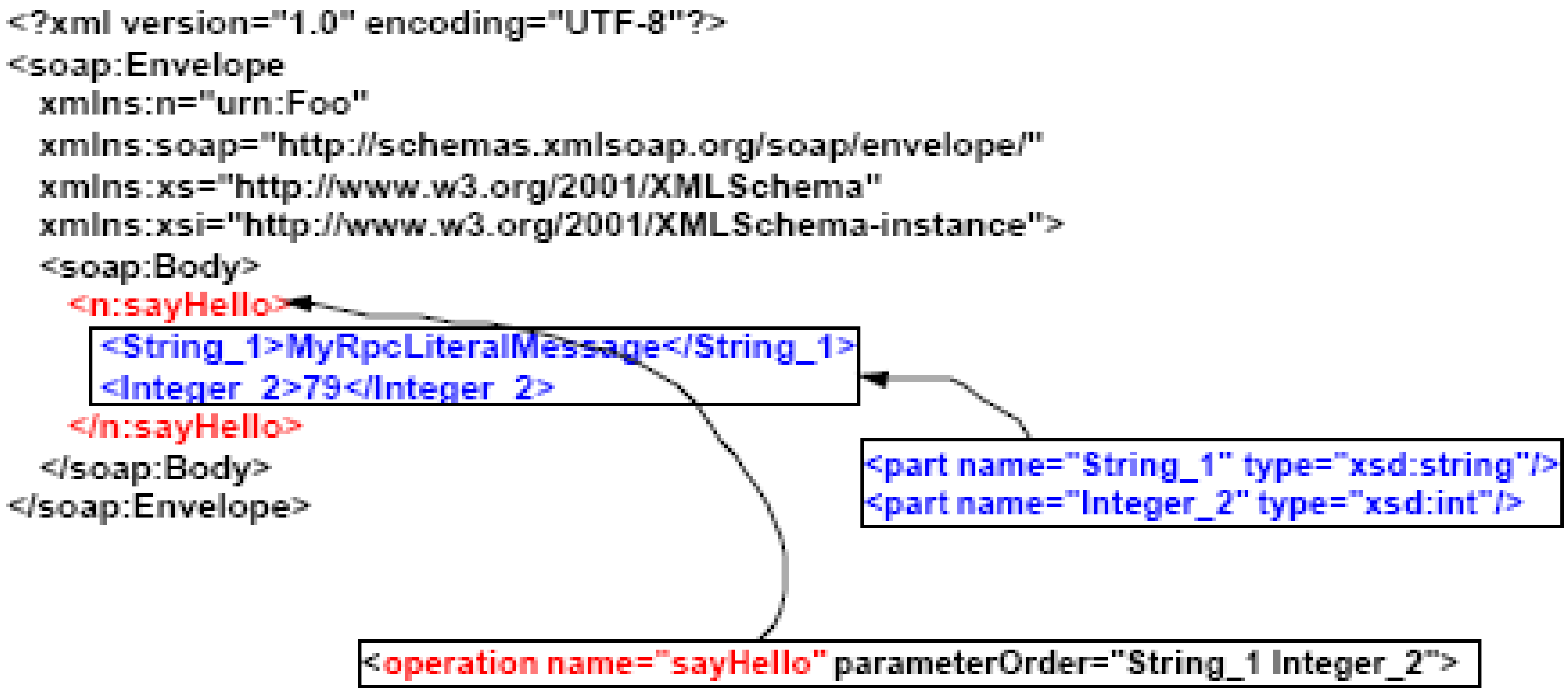
- WSDL document
  - The operation name of WSDL document is used to name the wrapper element (immediate child element under <soap:Body> element)
  - Each part is a parameter or a return value and appears inside a wrapper element within the <soap:Body>

- SOAP message:
  - Contents of the Body are formatted as a struct
  - Parts are arranged in the same order as the parameters of the call

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="MyHelloService"
     targetNamespace="urn:Foo">
  <types/>
  <message name="HelloIF_sayHello">
   <part name="String_1" type="xsd:string"/>
   <part name="Integer_2" type="xsd:int"/></message>
  <message name="HelloIF_sayHelloResponse">
   <part name="result" type="xsd:string"/></message>
  <portType name="HelloIF">
   <operation name="sayHello" parameterOrder="String_1 Integer_2">
    <input message="tns:HelloIF_sayHello"/>
    <output message="tns:HelloIF_sayHelloResponse"/></operation></portType>
  <binding name="HelloIFBinding" type="tns:HelloIF">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
   <operation name="sayHello">
    <input>
      <soap:body use="literal" namespace="urn:Foo"/></input>
    <output>
      <soap:body use="literal" namespace="urn:Foo"/></output>
    <soap:operation soapAction=""/></operation>
  </binding>
  <service name="MyHelloService">
   <port name="HelloIFPort" binding="tns:HelloIFBinding">
    <soap:address xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    location="http://localhost:8080/hello-jaxrpc/hello"/></port></service></definitions>
```

operation name

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name='HelloService' targetNamespace='http://hello.ws.jp.com/' xmlns='http://schemas.xmlsoap.org/wsdl/'
        xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' xmlns:tns='http://hello.ws.jp.com/' xmlns:xsd='http://www.w3.org/2001/
        XMLSchema'>
 <types/>
 <message name='HelloIF_sayHello'>
  <part name='String_1' type='xsd:string'/>
 </message>
 <message name='HelloIF_sayHelloResponse'>
  <part name='result' type='xsd:string'/>
 </message>
 <portType name='HelloIF'>
  <operation name='sayHello' parameterOrder='String_1'>
   <input message='tns:HelloIF_sayHello'/>
   <output message='tns:HelloIF_sayHelloResponse'/>
  </operation>
 </portType>
 <binding name='HelloIFBinding' type='tns:HelloIF'>
  <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http'/>
  <operation name='sayHello'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body namespace='http://hello.ws.jp.com/' use='literal'/>
   </input>
   <output>
    <soap:body namespace='http://hello.ws.jp.com/' use='literal'/>
   </output>
  </operation>
 </binding>
 <service name='HelloService'>
  <port binding='tns:HelloIFBinding' name='HelloIFPort'>
   <soap:address location='REPLACE_WITH_ACTUAL_URL'/>
  </port>
 </service>
</definitions>
```



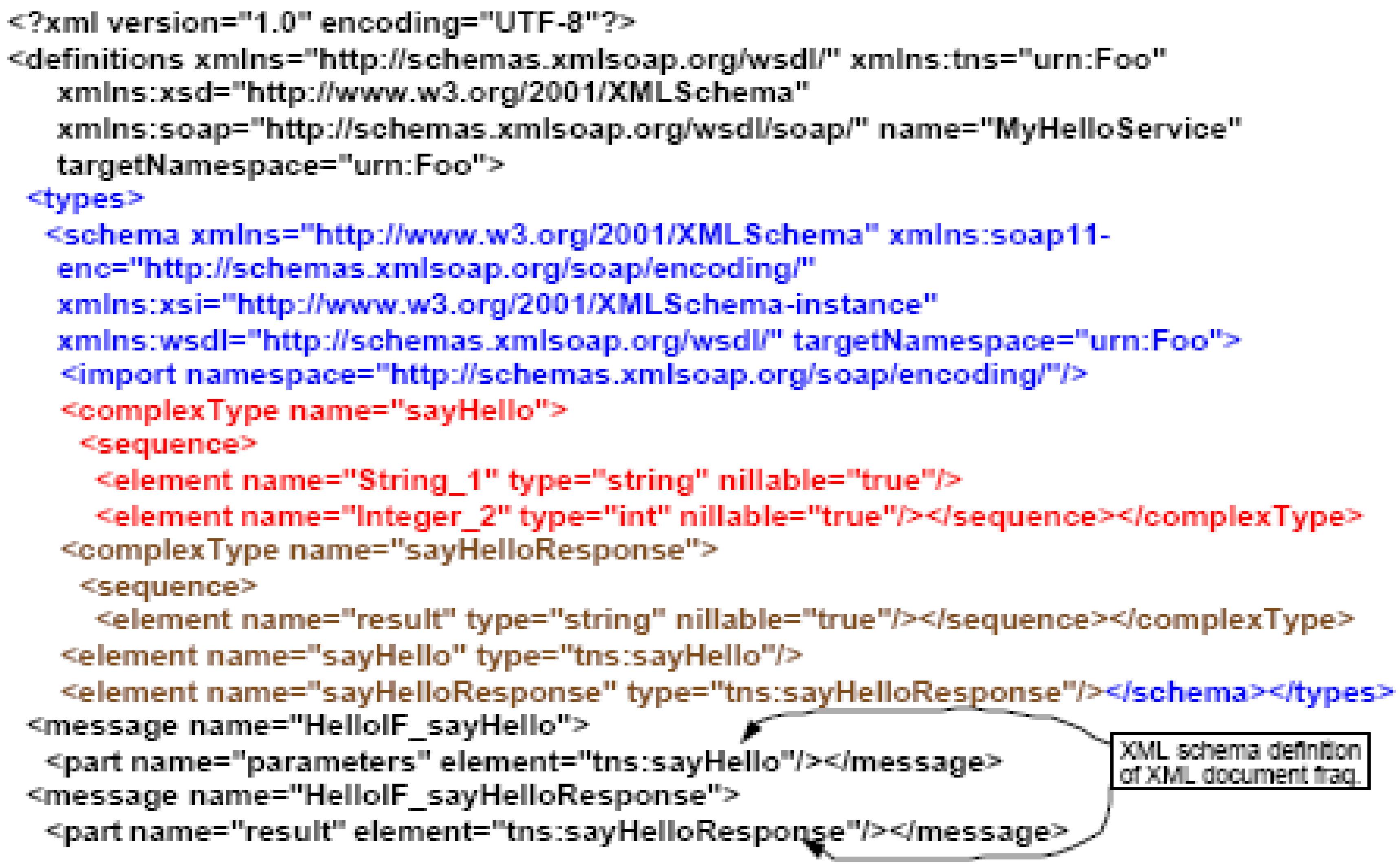
```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
   xmlns:n="urn:Foo"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soap:Body>
      <n:sayHello>
        <String_1>MyRpcLiteralMessage</String_1>
        <Integer_2>79</Integer_2>
      </n:sayHello>
   </soap:Body>
</soap:Envelope>
```

```
<part name="String_1" type="xsd:string"/>
<part name="Integer_2" type="xsd:int"/>
```

```
<operation name="sayHello" parameterOrder="String_1 Integer_2">
```

soap:body for Document style

- WSDL document:
  - Each <message> has single <part> element
  - The element attribute of <part> refers to schema definition of XML document fragment, which is defined inside <types>
- SOAP message:
  - SOAP Body element contains an XML document fragment (document)
- Ex) Purchase order XML document fragment
  - – There are no wrappers
- For document stype, each message has a single <part> element. And the element attribute of the <part> element defines schema definition of XML document fragment.



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="MyHelloService"
    targetNamespace="urn:Foo">
  <types>
   <schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:soap11-
    enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="urn:Foo">
    <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
    <complexType name="sayHello">
     <sequence>
      <element name="String_1" type="string" nillable="true"/>
      <element name="Integer_2" type="int" nillable="true"/></sequence></complexType>
    <complexType name="sayHelloResponse">
     <sequence>
      <element name="result" type="string" nillable="true"/></sequence></complexType>
    <element name="sayHello" type="tns:sayHello"/>
    <element name="sayHelloResponse" type="tns:sayHelloResponse"/></schema></types>
  <message name="HelloIF_sayHello">
   <part name="parameters" element="tns:sayHello"/></message>
  <message name="HelloIF_sayHelloResponse">
   <part name="result" element="tns:sayHelloResponse"/></message>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:tns="urn:Foo"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
     <tns:sayHello>
       <String_1>MyDocLiteralMessage</String_1>
       <Integer_2>78</Integer_2>
     </tns:sayHello>
   </soap:Body>
</soap:Envelope>
```

```
<complexType name="sayHello">
        <sequence>
          <element name="String_1" type="string" nillable="true"/>
          <element name="Integer_2" type="int" nillable="true"/></sequence></complexTy
```

use attribute of soap:body

- *use="literal|encoded"*
- *literal*
  - parts define the concrete schema of the message
  - XML document fragment can be validated against its XML schema
- *encoded*
  - Indicates whether the message parts are encoded using some encoding rules

use attribute of soap:body

- use=”literal”
  - each part references a concrete schema definition using either the *element* or *type* attribute (WS-I profile says use *element*)
  - *element* attribute
- Document style: the element referenced by the part will appear directly under the Body element
- RPC style: the element referenced by the part will appear under an accessor element named after the message part
  - *type* attribute
- the type referenced by the part becomes the schema type of the enclosing element

use attribute of *soap:body*

- use=”encoded”
  - each message part references an abstract type using the *t ype* attribute
  - abstract types are used to produce a concrete message by applying an encoding specified by the *encodingStyle* attribute
  - part names, types and value of the namespace attribute are all inputs to the encoding

Possible *Style/Use* Combinations

- style=”rpc” and use=”encoded”
- style=”rpc” and use=” literal”
- style=”document” and use=”encoded”
- style=”document” and use=”literal”