

**X
M
P
M**

Web Services

Dr. Yuhong Yan
NRC-IIT-Fredericton
Internet logic

OWL

RuleML

jDREW

UDDI

WSDL

SOAP

BPEL

Simple Object Access Protocol

SOAP

SOAP

SOAP (Simple Object Access Protocol)

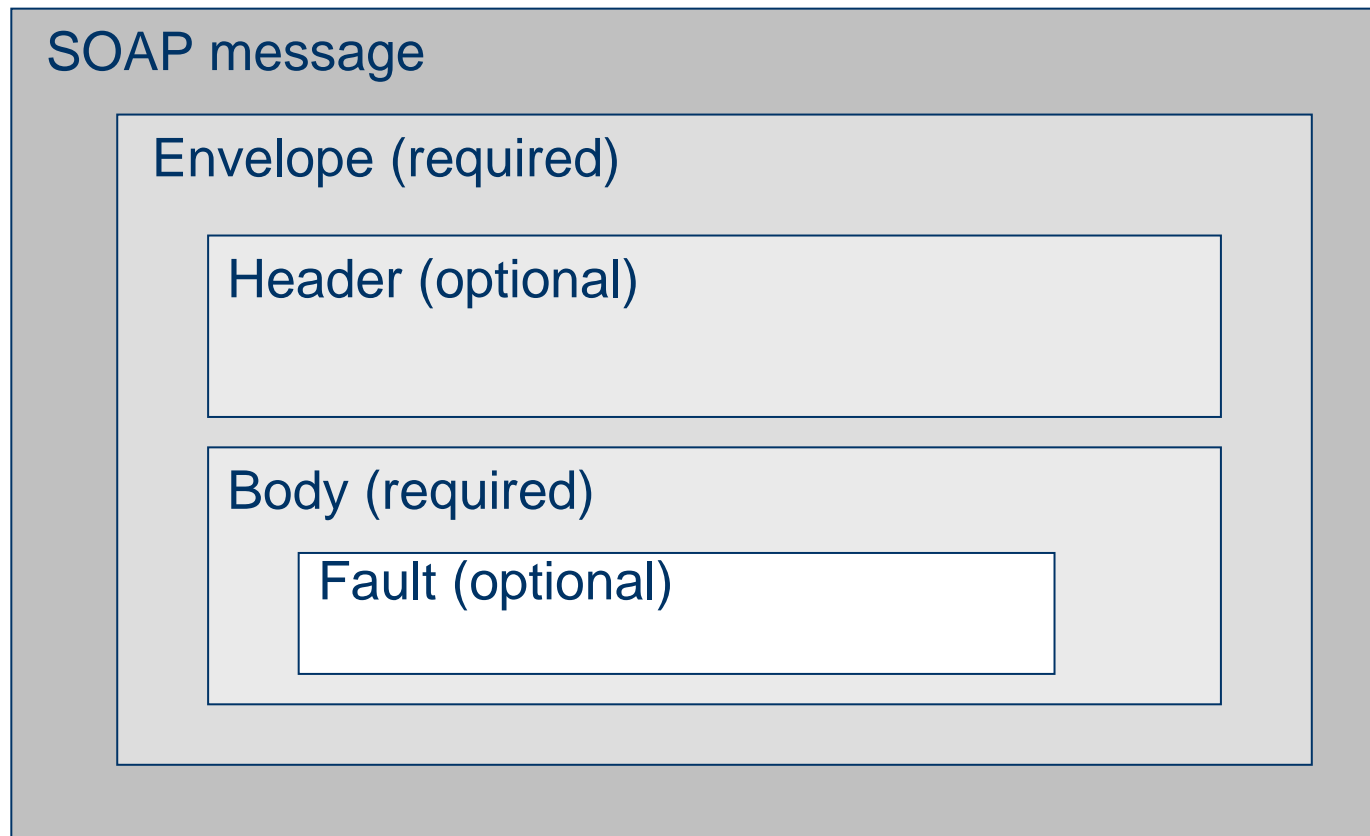
SOAP is a lightweight protocol for **exchange of information** in a decentralized, distributed environment.

- **XML based** protocol binds various internet protocols, like HTTP, SMTP, FTP, etc.
- Interoperability to Web Services

Why SOAP

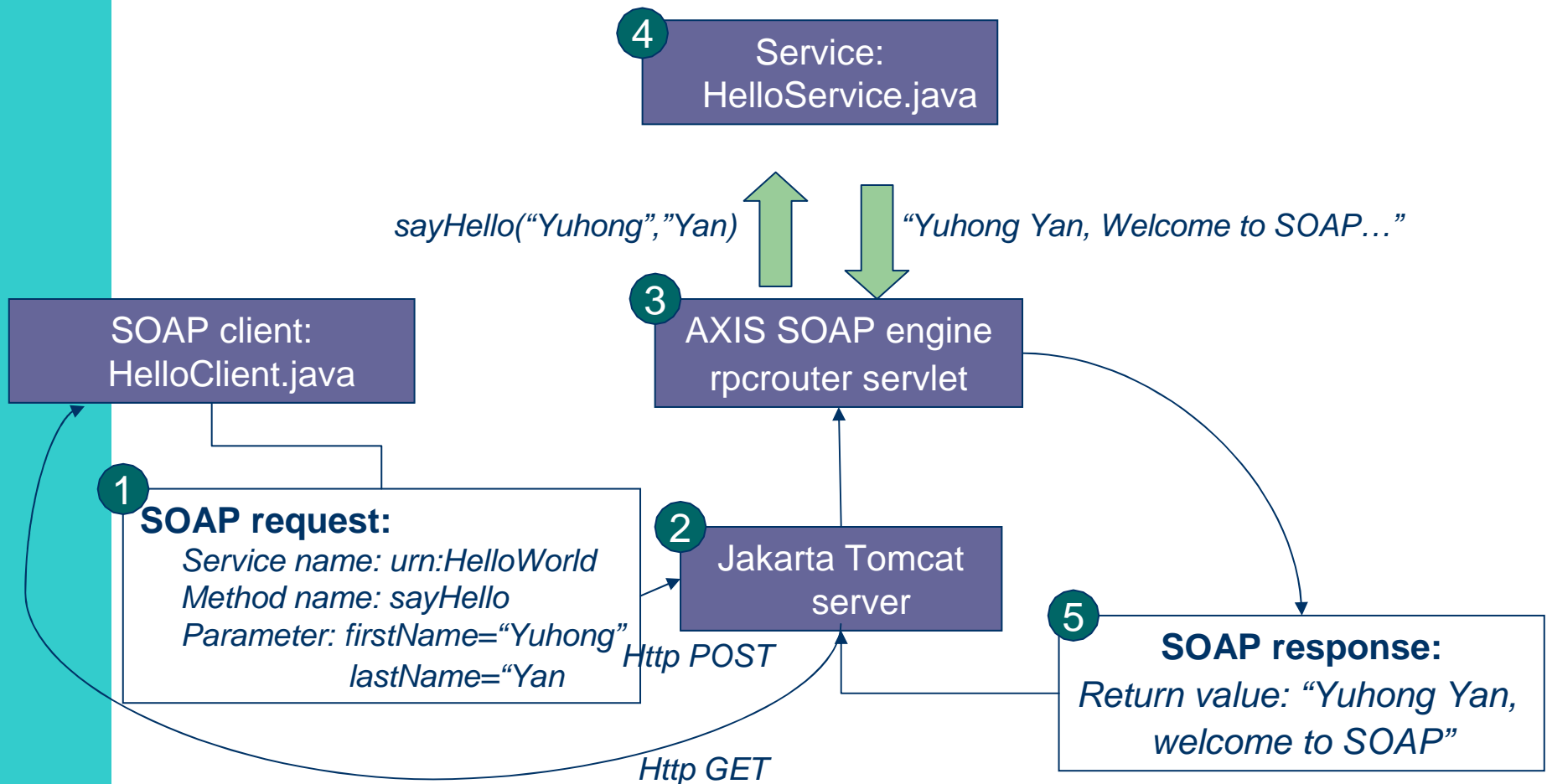
- inter-application communication between systems written in arbitrary languages, across the Internet.
- An XML-based protocol for exchanging messages over Internet transport protocols, like HTTP, SMTP, FTP, etc.
- SOAP is platform-independent.

Inside SOAP



From Etban Cerami, "Web Services Essentials", P53, figure 3-2

Apache SOAP architecture



From Etban Cerami, "Web Services Essentials", p69,. Fig 4-3

ICEC 2006 Tutorial on SOA

13-Aug-06

Anatomy of HelloWorld

- Server side code
- Client side code
- SOAP request
- SOAP response

HelloWorldService.java

```
package samples>HelloWorld;

public class HelloWorldService
{
    public String sayHello(String firstName, String lastName) throws Exception
    {
        String aString = firstName + " " + lastName + ", welcome to SOAP Web
        Service World!";
        return aString;
    }

    public String addString(String symbol, String dataType) throws Exception
    {
        String aString = symbol + dataType;
        return aString;
    }
}
```

TestClient

```
public class TestClient
{
    public static void main(String args[])
    {
        try
        {
            Options opts = new Options( args );
            args = opts.getRemainingArgs();

            Service service = new Service();
            Call call = (Call) service.createCall();
        }
    }
}
```

} *Parse
arguments*

} *Prepare to call
remote service*

TestClient.java (2)

Set the URL of the remote service

```
call.setTargetEndpointAddress( new java.net.URL(opts.getURL()) );
```

Set the URL of the remote service

```
if( args[0].equals("1") )
```

```
    call.setOperationName( new QName("urn:HelloWorld", "sayHello")); ←
```

```
else
```

Service Name

Service Method

```
    call.setOperationName( new QName("urn:HelloWorld", "addString")); ←
```

```
call.addParameter( "p1", XMLType.XSD_STRING,
    ParameterMode.IN );
```

```
call.addParameter( "p2", XMLType.XSD_STRING,
    ParameterMode.IN );
```

Add Paras for the remote methods

Paras Name

IN/OUT/INOUT

XML data type

TestClient.java (3)

```
call.setReturnType( XMLType.XSD_STRING );
```

} *Define the
return value*

```
call.setUsername( opts.getUser() );  
call.setPassword( opts.getPassword() );
```

} *Security
options*

```
String res = (String) call.invoke( new Object[] { args[1], args[2] } );
```

} *Invoke
Service*

```
System.out.println( "Return is: " + res );
```

```
}
```

```
catch( Exception e )
```

```
{
```

```
    e.printStackTrace();
```

```
}
```

```
}
```

```
}
```

↖ *Pass two paras*

Request SOAP message

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:sayHello
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:ns1="urn:HelloWorld">
      <p1 xsi:type="xsd:string">Yuhong</p1>
      <p2 xsi:type="xsd:string">Yan</p2>
    </ns1:sayHello>
  </soapenv:Body>
</soapenv:Envelope>
```

Remote method name

Parameter name

Name of the web service

Value of the parameter

Parameter type. need to match WSDL.

Response SOAP message

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:sayHelloResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:ns1="urn:HelloWorld">
      <ns1:sayHelloReturn xsi:type="soapenc:string"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
        Yuhong, Yan, welcome to SOAP Web Service World!
      </ns1:sayHelloReturn>
    </ns1:sayHelloResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP Encoding

- Scalar Type: map to simple types in XML Schema
`<return xsi:type="xsd:double" > 54.99 </return>`

Primitives: the mapping table

xsd:base64Binary	byte[]
xsd:boolean	boolean
xsd:byte	byte
xsd:dateTime	java.util.Calendar
xsd:decimal	java.math.BigDecimal
xsd:double	double
xsd:float	float
xsd:hexBinary	byte[]
xsd:int	int
xsd:integer	java.math.BigInteger
xsd:long	long
xsd:QName	javax.xml.namespace.QName
xsd:short	short
xsd:string	java.lang.String

Compound Type

- Arrays: map to array in XML schema

```
<return ...>
```

```
  <item xsi:type="xsd:double" > 54.99 </item>
```

```
  <item xsi:type="xsd:double" > 19.99 </item>
```

```
</return>
```

- Structures:

```
<return ...>
```

```
  <name xsi:type="xsd:string"> Elia and Louis </name>
```

```
  <price xsi:type="xsd:double" > 19.99 </item>
```

```
</return>
```

SOAP Extensions

- Authentication
- Authorization
- Reliability
- Correlation
- Transaction

Header (optional)

- For authentication, transaction management, and payment authorization
- Two defined attributes
 - Actor attribute: the chained node
 - MustUnderstand attribute: force the recipient to process the element, if not understandable, return a fault

Header (optional) (2)

```
<SOAP-ENV:Header>
```

```
  <ns1:PaymentAccount
```

```
    xmlns:ns1="urn:ecerami"
```

```
    ENV:mustUnderstand="true">
```

```
      orsenigo473
```

```
    </ns1:PaymentAccount>
```

```
</SOAP-ENV:Header>
```

SOAP-

P54. the soapheader

Fault (optional)

- **faultCode**
 - SOAP-ENV:VersionMismatch
 - SOAP-ENV:MustUnderstand
 - SOAP-ENV:Client (non existing methods)
 - SOAP-ENV:Server (not able to access DB)
- **faultString**
- **faultActor**
- **detail**

Fault (optional)-2

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode xsi:type="xsd:string">SOAP-ENV:Client</faultcode>
      <faultstring xsi:type="xsd:string">
        Failed to locate method (ValidateCreditCard) in class
        (examplesCreditCard) at /usr/local/ActivePerl-5.6/lib/
        site_perl/5.6.0/SOAP/Lite.pm line 1555.
      </faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Web Service Description Language

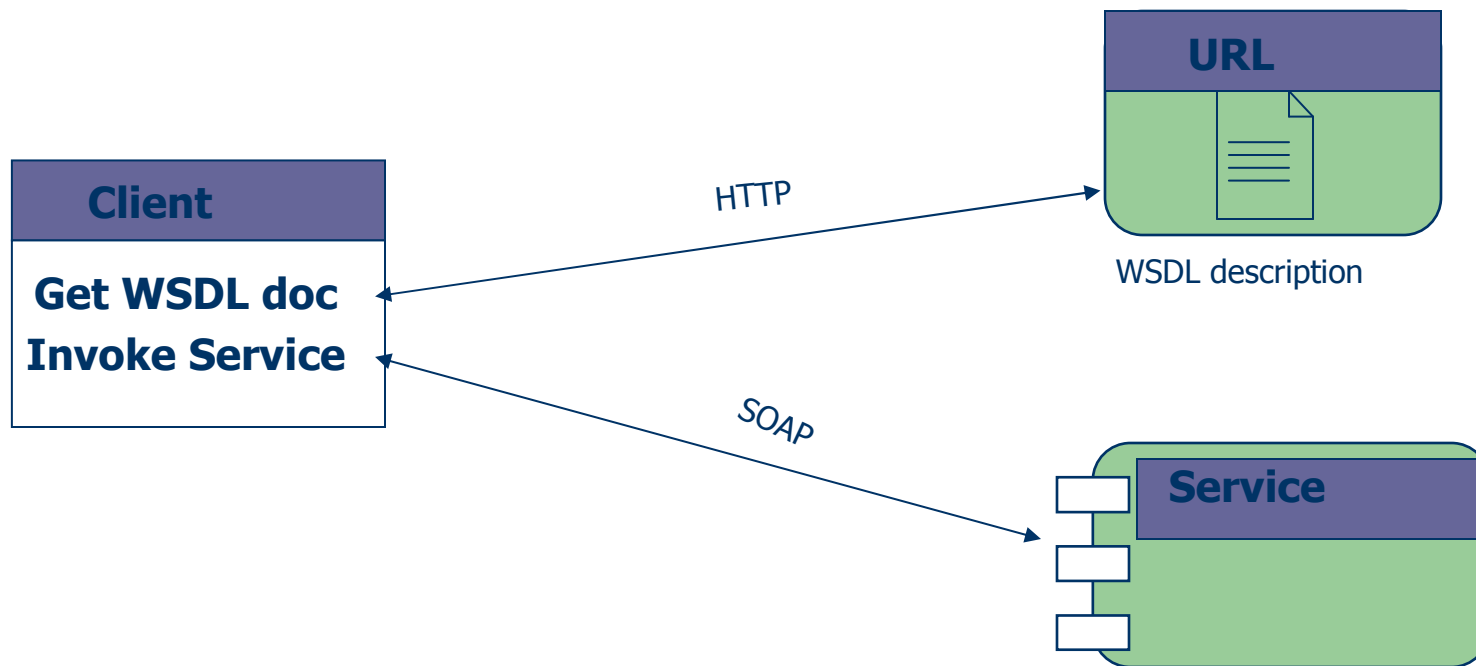
WSDL

W
S
D
L

WSDL:

- A contract between requestor and provider
- Platform and language-independent
- Describe SOAP services
- Automatic tool to generate client and sever code

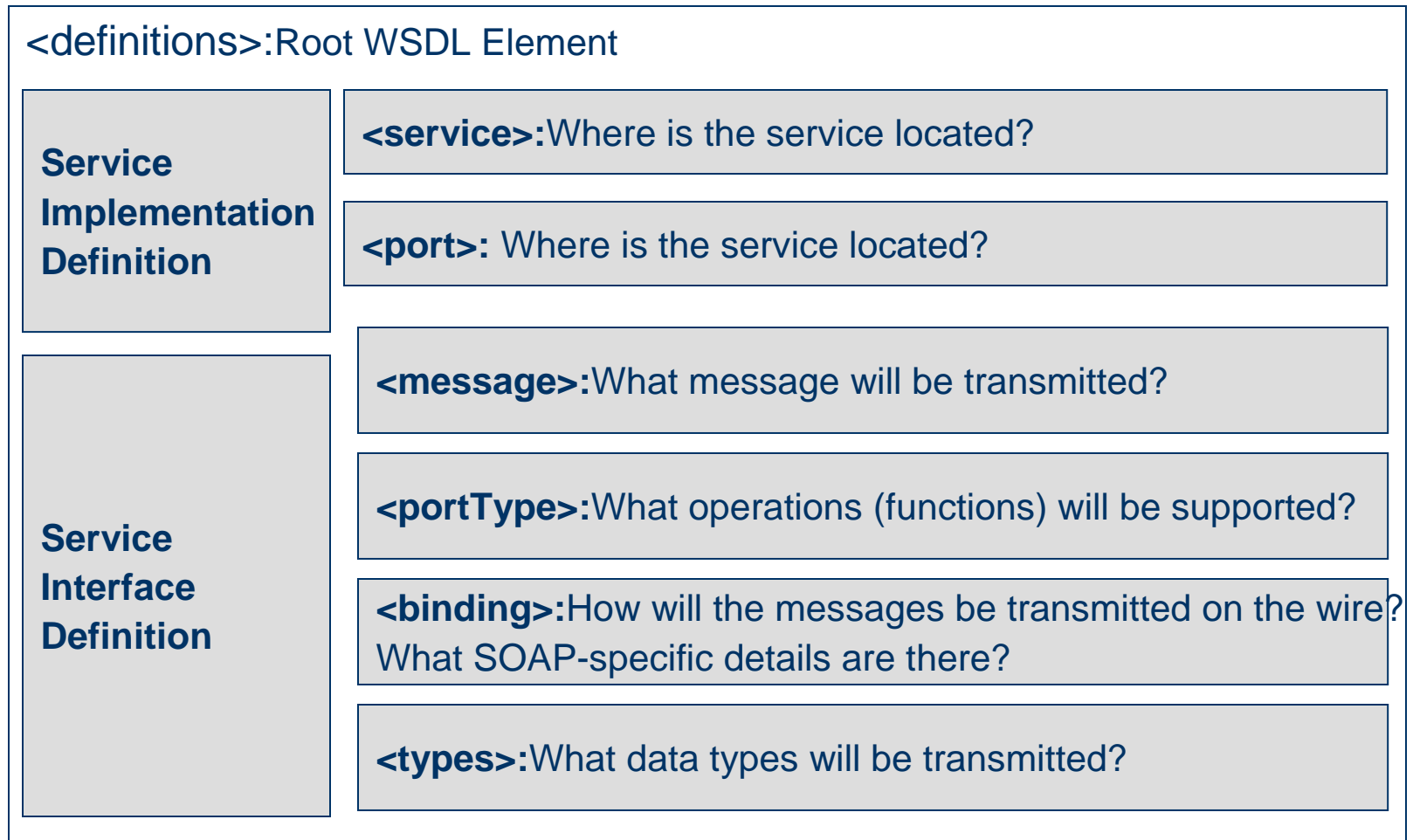
Interoperability Using WSDL



WSDL: Describe a web service

- **Interface information** describing all publicly available functions
- **Data type information** for all message requests and message responses
- **Binding information** about the transport protocol to be used
- **Address information** for locating the specified services

Structure of WSDL




From Etban Cerami, "Web Services Essentials", P121 fig 6-1

An example of WSDL: HelloWorld

```
< sdl:definitions
targetNamespace="http://localhost:8080/axis/services/urn:Hello
World" xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://localhost:8080/axis/services/urn:HelloWorld"
xmlns:intf="http://localhost:8080/axis/services/urn:HelloWorld"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://www.w3.org/1999/XMLSchema"
xmlns:tns2="http://www.w3.org/2003/05/soap-encoding"
xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
xmlns:wSDLsoap="http://schemas.xmlsoap.org/wSDL/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

<Service> : where the service is located

```
<wsdl:service name="HelloWorldServiceService">  
  <wsdl:port  
    binding="impl:urn:HelloWorldSoapBinding"  
    name="urn:HelloWorld">  
    <wsdlsoap:address  
      location="http://localhost:8080/axis/services/urn:HelloWorld" />  
    </wsdl:port>  
</wsdl:service>
```



The service location:
the endpoint of the

<Binding>: bind messages to operations

```
< /wsdl:binding name="urn:HelloWorldSoapBinding"
  type="impl:HelloWorldService">
  <wsdlsoap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="sayHello">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="sayHelloRequest">
  <wsdlsoap:body
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="http://HelloWorld.samples" use="encoded" />
  </wsdl:input>
- <wsdl:output name="sayHelloResponse">
  <wsdlsoap:body
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    namespace="http://localhost:8080/axis/services/urn:HelloWorld"
    use="encoded" />
  </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

Use SOAP HTTP
protocol

<portType> : operations supported

Input Message

```
<wsdl:portType name="HelloWorldService">  
  <wsdl:operation name="sayHello"  
    parameterOrder="in0 in1">  
    <wsdl:input message="impl:sayHelloRequest"  
      name="sayHelloRequest" />  
    <wsdl:output  
      message="impl:sayHelloResponse"  
      name="sayHelloResponse" />  
  </wsdl:operation>  
</wsdl:portType>
```

Output Message

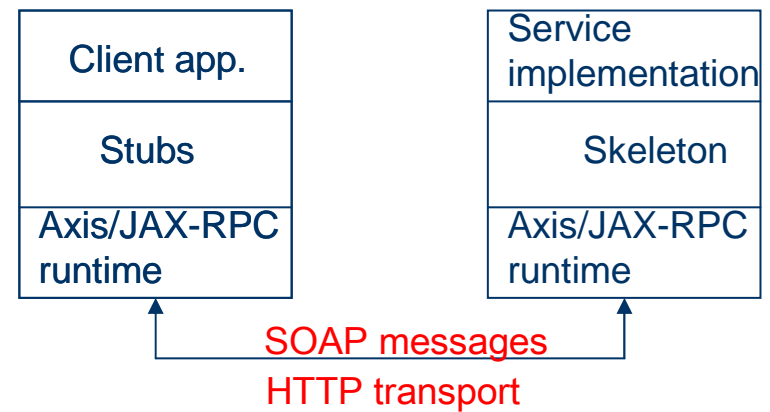
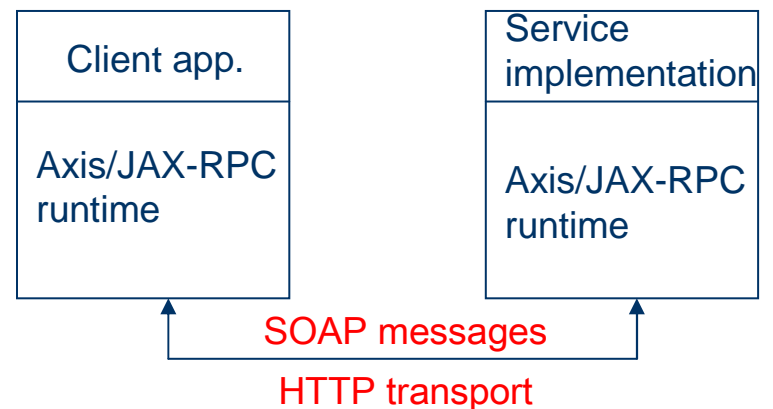
<message>: define content of message

```
<wsdl:message name="sayHelloRequest">
  <wsdl:part name="in0" type="tns:string" />
  <wsdl:part name="in1" type="tns:string" />
</wsdl:message>

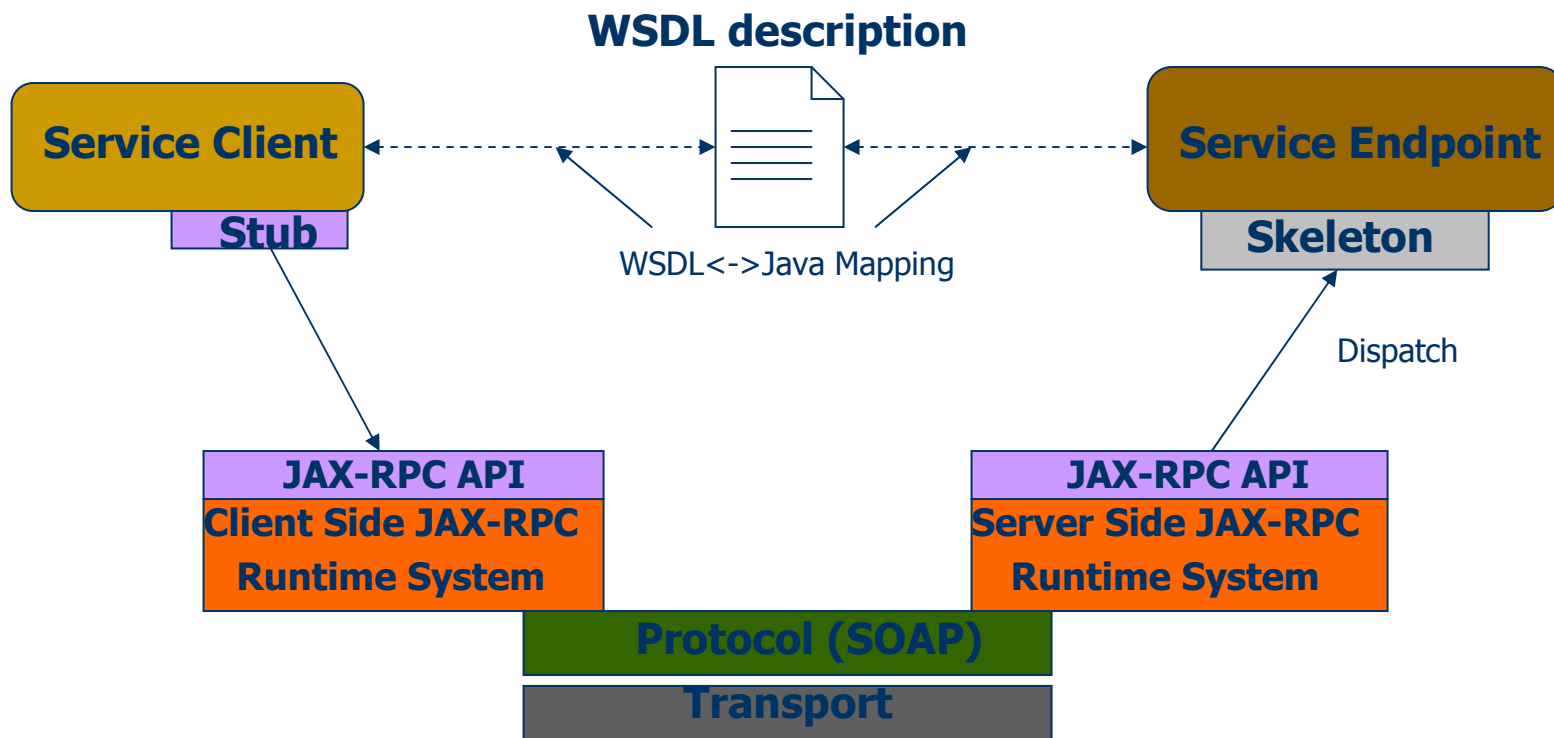
<wsdl:message name="sayHelloResponse">
  <wsdl:part name="sayHelloReturn"
    type="tns:string" />
</wsdl:message>
```

Three ways for Client to invoke a Service

- Dynamic invocation interface (DII) (bottom right).
 - No stubs/ties; WSDL not necessary.
 - More coding
 - Flexible.
- Using static stubs (top right).
 - Tool creates stubs and ties c.f. RMI; client operates on proxy (stub).
 - Less code than DII.
- Dynamic proxy.
 - No stubs, client creates proxy class at runtime from WSDL.
 - More portable than static stubs.



Stub-Skeleton Architecture



WSDL2Java

WSDL clause	Java class(es) generated
For each entry in the type section	A java class
	A holder if this type is used as an inout/out parameter
For each portType	A java interface
For each binding	A stub class
For each service	A service interface
	A service implementation (the locator)

Web Services vs. Middlewares

	Web Service	RMI	CORBA	DCOM
Transport Protocol	Http	JRMP	IIOP	ORPC
Data Binding	XML schema (allow customized data type)	Primitive, Serialized objects	IDL (primitive and structure)	MS IDL
Compliant prog. Langs.	Any (with XML parser, SOAP composition)	java	Any (with IDL mapping standards)	Many (C++, Java, VB, etc.)

	Web Services	RMI	CORBA	DCOM
Interface Description	WSDL	Interface of server objects	IDL	MS IDL
Remote Call	By SOAP message	Get references of server objects	Get reference of server object	Get Pointer of server objects
Routine	Stub Proxy DII	Client: stub or proxy Server: skeleton	Client: stub or proxy Server: skeleton	Client: proxy Server: stub

Web Service Discovery

Discovery

Discovery

Three Approaches

- The registry approach
- The index approach
- The peer-to-peer approach

UDDI: Universal Description, Discovery, and Integration

- White pages
- Yellow pages
- Green pages

White pages

- Information on a **business** itself
 - Locations of the business
 - Contact names
 - Unique identifiers, e.g. tax IDs
 - URL
- Associated with <businessEntity> element

Yellow pages

- Categorized information about the **services** provided by a business.
- Categorization: assigning one or more taxonomies to the business
 - E.g. a service is categorized as an “online store” and as an “Music Store”
- Associated with `<businessService>` element

Green pages

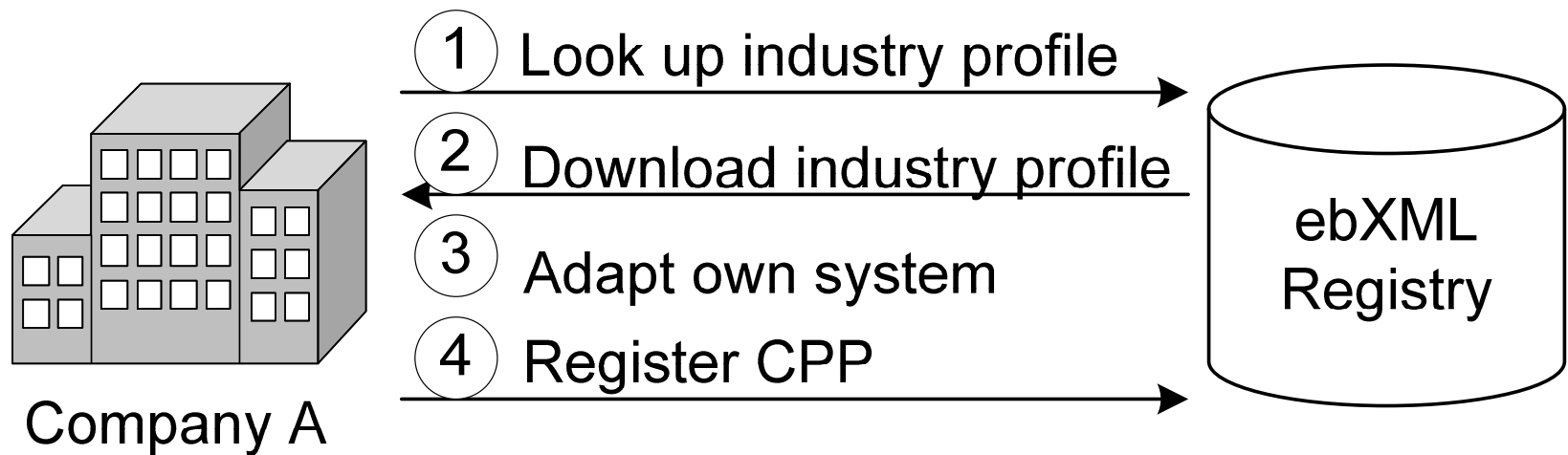
- **Technical information** about a service which is offered by a business.
 - Service location (URL)
 - Category
 - Specification of the service
- Associated with `<businessService>` and `<bindingTemplate>` elements

Web Service vs. ebXML

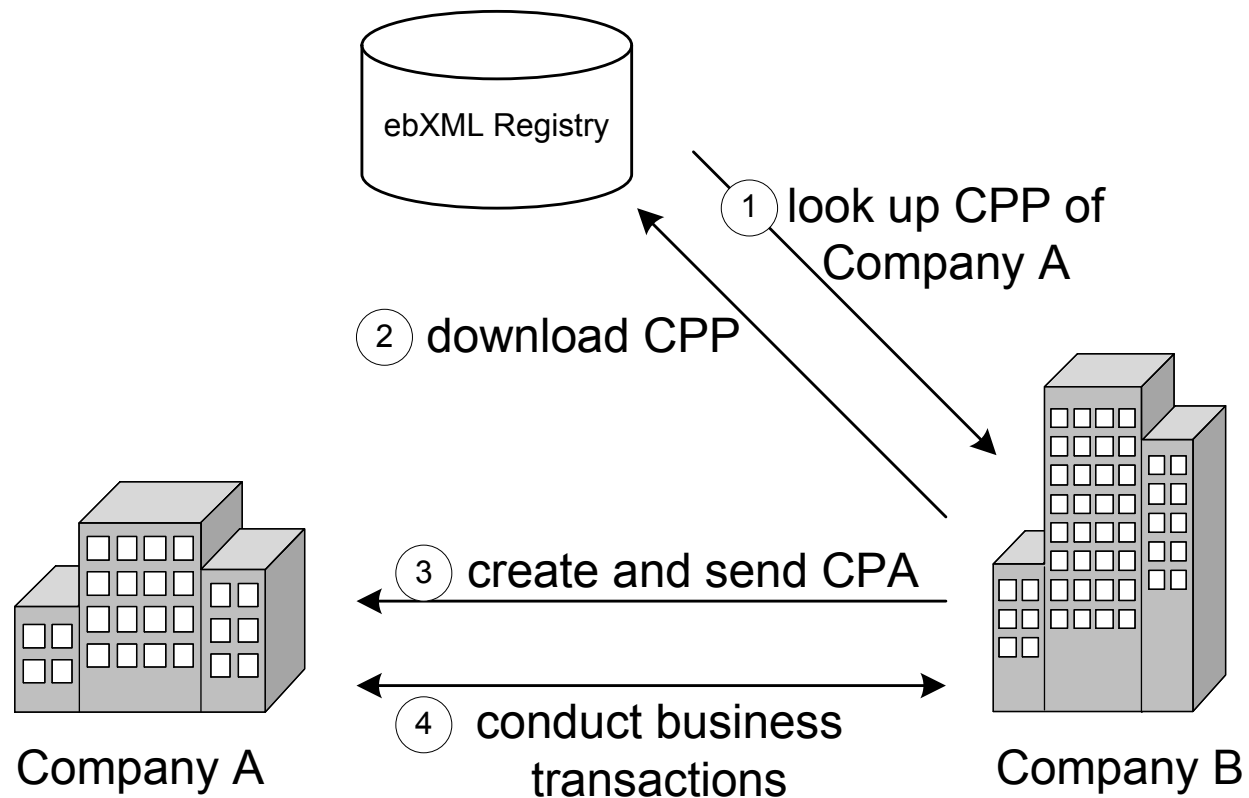
ebXML

ebXML

Implementation phase of ebXML

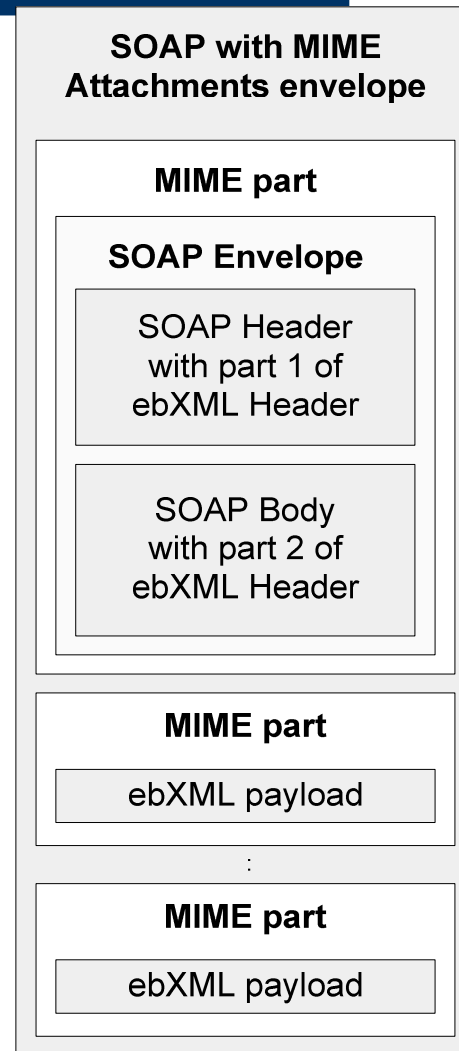
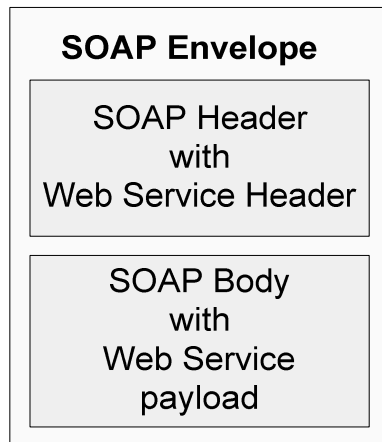


Runtime phase in ebXML



SOAP in ebXML

SOAP structure for Web Service



SOAP structure for ebXML

ebXML components

- Business documents:
 - Core components
- Business transactions
- Trade partner agreements
 - CPP
 - CAP
- Business registration