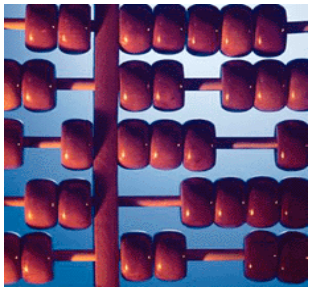
Web Services 2

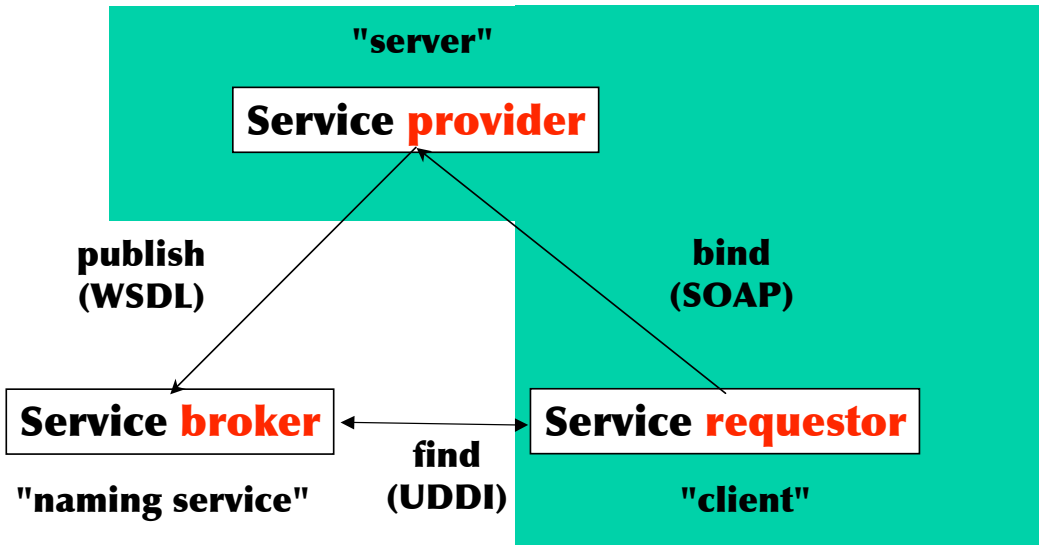
Lecture 9



Outline

- SOAP (messaging)
- WSDL (service description)
- WS-Interoperability (WS-I)
- UDDI (registry)
- Some practical aspects
- REST

Reminder: Web Service Architecture



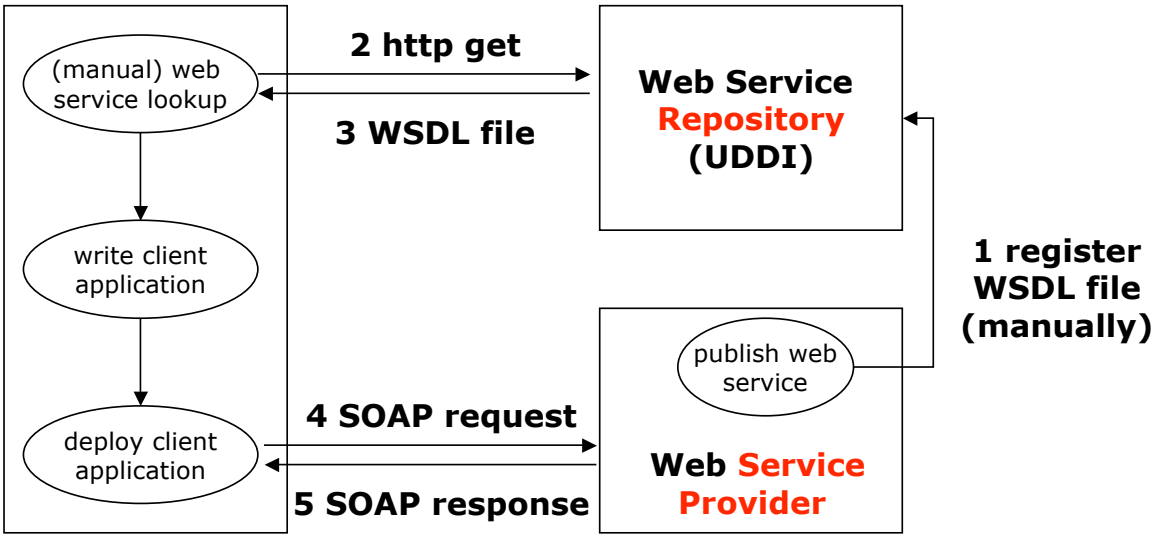
Message Exchange Patterns

- Request
  - Simple request without response
- Request-Response
  - Client sends request and server responds on same port
  - Most common

Web Evolution

Technology	TCP/IP	HTML	XML
Purpose	Connectivity	Presentation	Programmability
Applications	E-Mail, FTP…	Web Pages	Web Services
Outcome	Create the Web	Browse the Web	Program the Web

Basic Web Service Usage Scenario

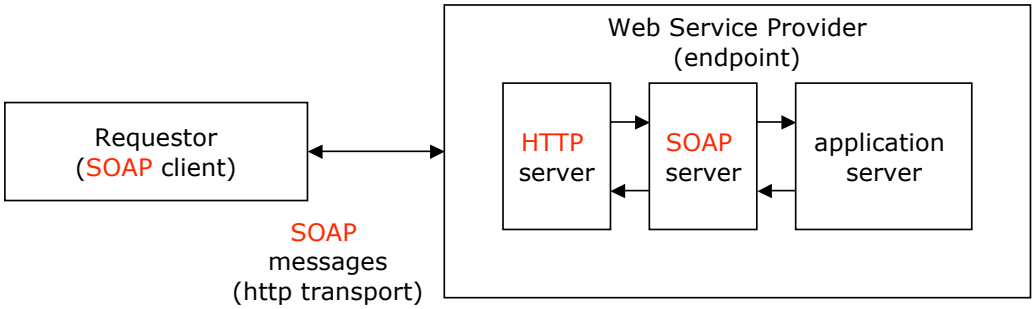


Web Service Stack

- A set of standards for implementing Web services

Publication and Discovery: **UDDI**	extends URI
Service Description: **WSDL**	extends HTML
Messaging: **SOAP**	extends HTTP
Transport: HTTP, SMTP, FTTP, …	

Web Services Implementation



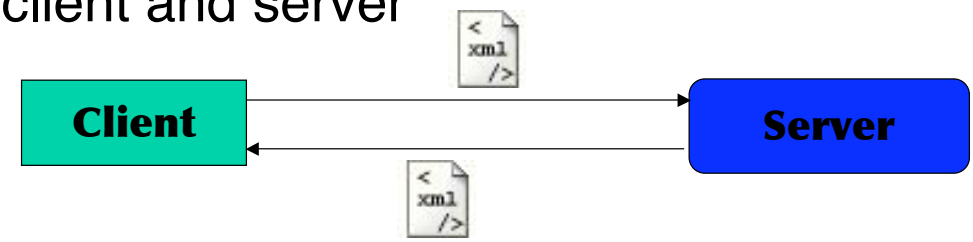
- Application Server (web service-enabled)
  - provides implementation of services and exposes it through WSDL/SOAP
  - implementation in Java, as EJB, as .NET (C#) etc.
- SOAP server
  - implements the SOAP protocol
- HTTP server
  - standard Web server
- SOAP client
  - implements the SOAP protocol on the client site

Example
Amazon Web Services

- www.amazon.com/gp/aws/landing.html
- Very large product database exposed through Web Services
- Lots of marketing using a certain technology
- Idea: let others figure out how to sell products for us
  - Associates program enables Web sites to link to Amazon.com and earn referral fees

SOAP in simple Words

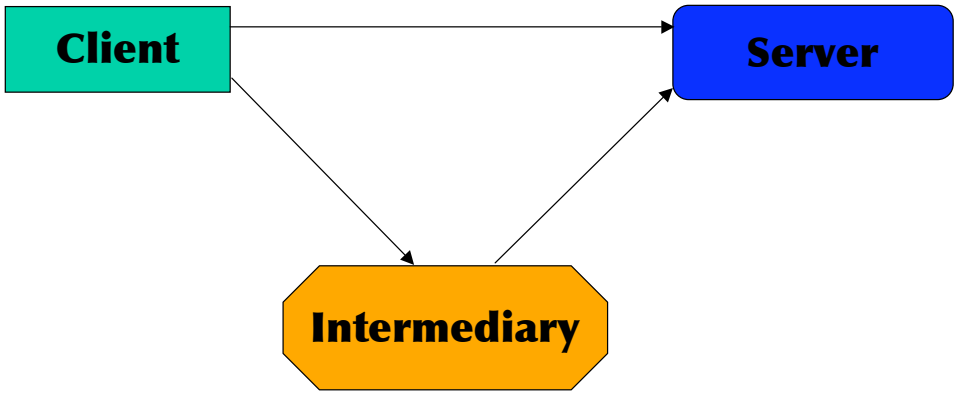
- Originally called “Simple Object Access Protocol”
- Defines the way how XML documents (messages) are exchanged between client and server



SOAP

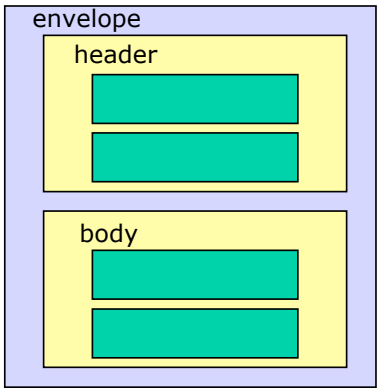
- Lightweight messaging framework based on XML
- Supports simple messaging and RPC (remote procedure calls)
- SOAP consists of
  - Envelope construct: defines the overall structure of messages
  - Encoding rules: define the serialization of application data types
  - SOAP RPC: defines representation of remote procedure calls and responses
  - Binding framework: binding to protocols such as HTTP, SMTP
  - Fault handling
- SOAP supports advanced message processing:
  - *forwarding intermediaries*: route messages based on the semantics of message
  - *active intermediaries*: do additional processing before forwarding messages, may modify message

Client-Server Interaction



SOAP Message

- SOAP messages consist of
  - Envelope: top element of XML message (*required*)
  - Header: general information on message such as security (*optional*)
  - Body: data exchanged (required)
- Header
  - elements are application-specific
  - may be processed and changed by *intermediaries* or recipient
- Body
  - elements are application-specific
  - processed by recipient only



Skeleton SOAP Message

```
<?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:Header>
  ...
  ...
</soap:Header>

<soap:Body>
  ...
  ...
  <soap:Fault>
    ...
    ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

Minimal SOAP/HTTP Request

- SOAP Header is optional so we skip it here

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?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.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

http://www.w3schools.com/soap/soap_example.asp

Minimal SOAP/HTTP Response

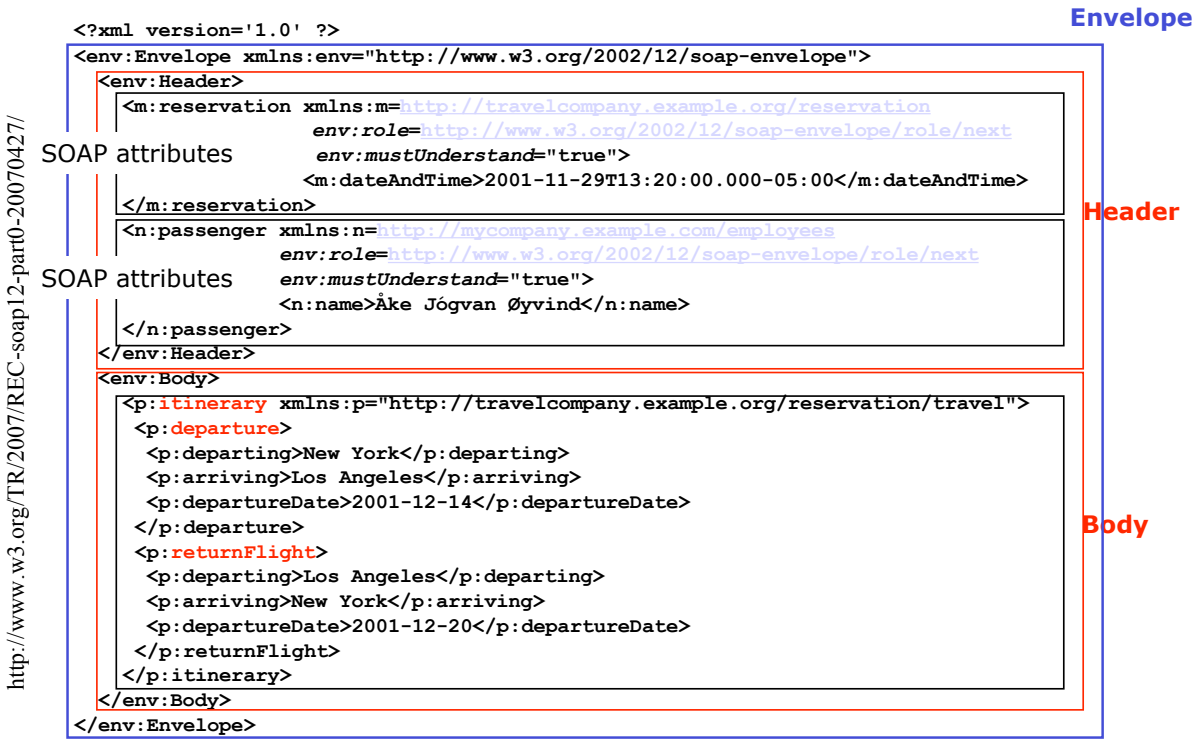
```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?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.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

http://www.w3schools.com/soap/soap_example.asp

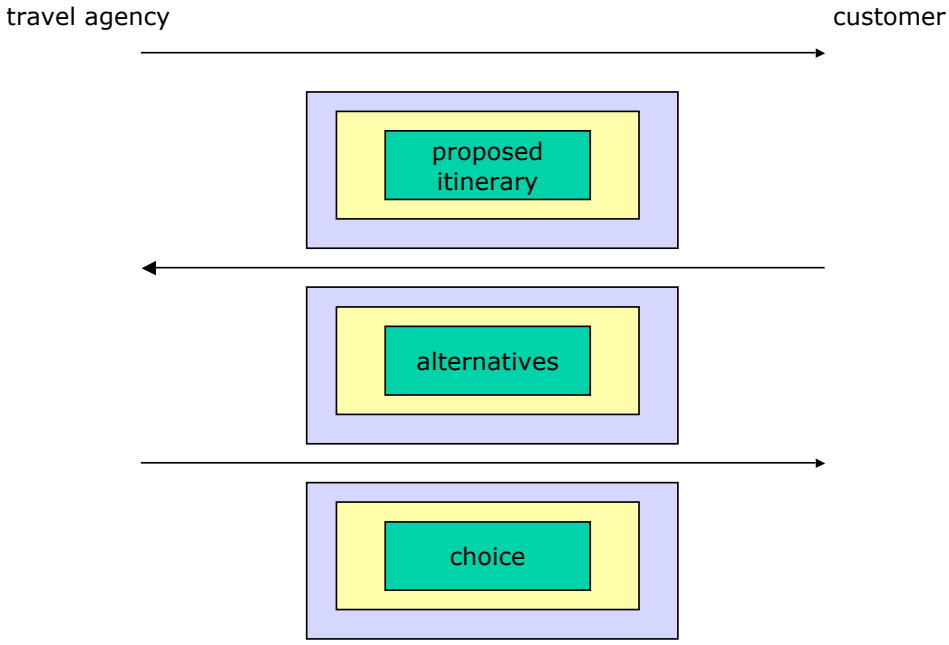
Java Class

```
public class TravelAgency {
  public int reservation(String date),
  public int passenger(String name),
  public int itinery(String departing,
                     String arriving,
                     String departureDate),
  public int returnFlight(String departing,
                          String arriving,
                          String departureDate)
}
```


Example: “Complex” SOAP Message



Conversational Message Exchanges in SOAP



SOAP RPC (Request)

- Encapsulate RPC into SOAP messages
  - procedure name and arguments
  - response (return value)
  - processing instructions (transactional RPC!)
- Example: **Request** message

```
<?xml version='1.0' ?>
    <env:Envelope xmlns:env="http://www.w3.org/2002/12/soap-envelope" >
    <env:Header>                                                    transaction information
      <t:transaction xmlns:t="http://thirdparty.example.org/transaction"
                     env:encodingStyle="http://example.com/encoding"
                     env:mustUnderstand="true" >5</t:transaction>
    </env:Header>                                                   TID
    <env:Body>                                                      method invocation
      <m:chargeReservationRequest env:encodingStyle="http://www.w3.org/2002/12/soap-encoding"
                        xmlns:m="http://travelcompany.example.org/">
        <m:reservation xmlns:m="http://travelcompany.example.org/reservation">
          <m:code>FT35ZBQ</m:code>                                  parameter 1
        </m:reservation>
        <o:creditCard xmlns:o="http://mycompany.example.com/financial">
          <n:name xmlns:n="http://mycompany.example.com/employees">
              Åke Jógvan Øyvind </n:name>
          <o:number>123456789099999</o:number>
          <o:expiration>2005-02</o:expiration>                      parameter 2
        </o:creditCard>
      </m:chargeReservationRequest>
    </env:Body>
    </env:Envelope>
```

SOAP RPC (Response)

- Example cntd.: **Response** message

```
<?xml version='1.0' ?>
   <env:Envelope xmlns:env="http://www.w3.org/2002/12/soap-envelope" >
   <env:Header>
     <t:transaction xmlns:t=http://thirdparty.example.org/transaction
                    env:encodingStyle=http://example.com/encoding
                    env:mustUnderstand="true">5</t:transaction>
   </env:Header>
   <env:Body>
     <m:chargeReservationResponse
                    env:encodingStyle=http://www.w3.org/2002/12/soap-encoding
                    xmlns:m="http://travelcompany.example.org/">
       <m:code>FT35ZBQ</m:code>
       <m:viewAt> http://travelcompany.example.org/reservations?code=FT35ZBQ
       </m:viewAt>
     </m:chargeReservationResponse>
   </env:Body>
   </env:Envelope>
```

method result

output parameters

SOAP Processing Model (1)

- Elements in the Header may carry SOAP-specific attributes controlling the message processing
  - attributes from namespace
    `http://www.w3.org/2002/12/soap-envelope`
  - role, mustUnderstand, relay, encodingStyle
- "**role**" attribute
  - if processing node matches role in header, it must process the header
  - special role "next": receiving node must be capable of processing header
  - special role "ultimateReceiver: receiving node must be capable of processing body
- "**mustUnderstand**" attribute
  - processing of header information is mandatory

SOAP Processing Model (2)

- "**relay**" attribute
  - header block must be relayed if it is not processed
- "**encodingStyle**" attribute
  - Indicates the encoding rules used to serialize parts of a SOAP messages
    - "http://www.w3.org/2003/05/soap-encoding"
      - **Base64**
      - **date**
      - **hexBinary** …
    - "http://example.org/encoding/"
    - "http://www.w3.org/2003/05/soap-envelope/encoding/none"

The Fault Element

- Carries an error message
- If present, must appear as a child of <Body>
- Must only appear once
- Has the following sub-elements:

Sub Element	Description
<faultcode>	A code for identifying the fault (VersionMismatch, MustUnderstand, Client, Server)
<faultstring>	A human readable explanation of the fault
<faultactor>	Information about who caused the fault to happen
<detail>	Holds application specific error information related to the Body element

Example: Fault Response

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:mh="http://www.Monson-Haefel.com/jwsbook/BookQuote" >
  <soap:Body>
    <soap:Fault>
     <faultcode>soap:Client</faultcode>
     <faultstring>
       The ISBN value contains invalid characters
     </faultstring>
     <faultactor>http://www.xyzcorp.com</faultactor>
     <detail>
       <mh:InvalidIsbnFaultDetail>
         <offending-value>19318224-D</offending-value>
         <conformance-rules>
          The first nine characters must be digits. The last
           character may be a digit or the letter 'X'. Case is
           not important.
         </conformance-rules>
       </mh:InvalidIsbnFaultDetail>
     </detail>
   </soap:Fault>
 </soap:Body>
</soap:Envelope>
```


http://www.awprofessional.com/articles/article.asp?p=169106&seqNum=6&rl=1

Protocol Binding

- Bindings to different protocols possible: HTTP, SMTP
- Different HTTP bindings: HTTP POST, HTTP GET
  - Standard: HTTP POST for request-response

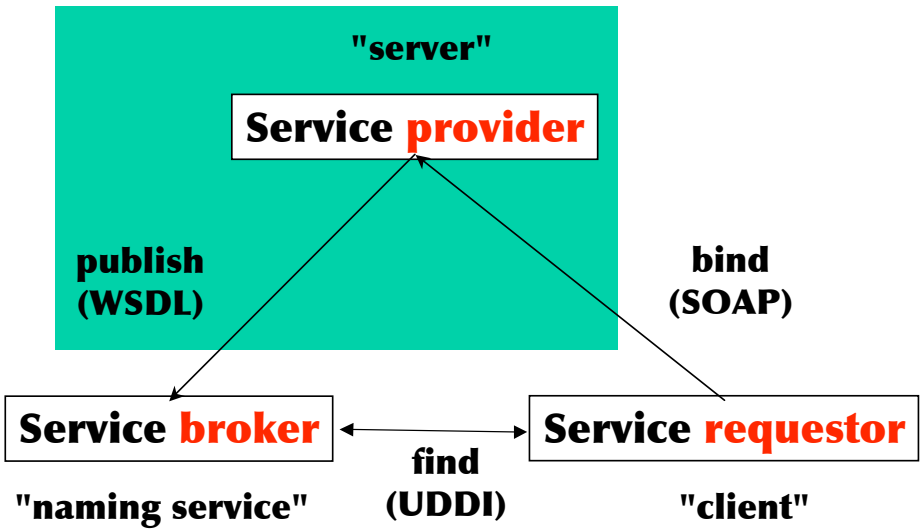
```
POST /Reservations?code=FT35ZBQ HTTP/1.1
Host: travelcompany.example.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/12/soap-envelope">
 …SOAP request message…
</env:Envelope>
```

HTTP POST request

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/12/soap-envelope">
… SOAP response message …
</env:Envelope>
```

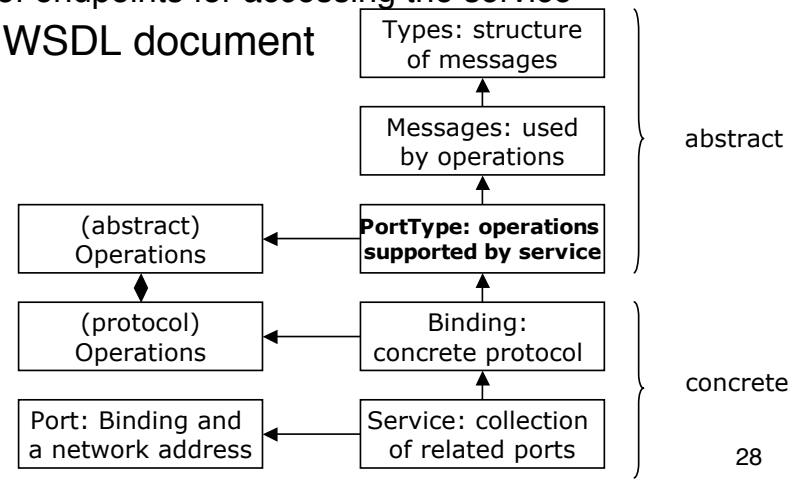
HTTP response

Reminder: Web Service Architecture



WSDL – Web Service Description Language

- Description of Web services in XML format
  - abstract description of operations and their parameters (messages)
  - binding to a concrete network protocol (e.g. SOAP)
  - specification of endpoints for accessing the service
- Structure of a WSDL document



Overview of Defining WSDL Services

1. Define in XML Schema the message types used when invoking the service: MT1, MT2 etc.
2. Define (named) messages by using these types, e.g.
   - message m1 has type MT1
   - message m2 has type MT2 etc.
3. Define Services that consist of one or more operations; each operation is implemented by the exchange of messages
   - service S offers operation O1; for executing O1 first send a request message m1, then a response message m2 is returned
4. Define a Binding B to a specific protocol, e.g. SOAP
   - service S is implemented in SOAP; the SOAP messages are constructed from the abstract messages m1 and m2 by, e.g. inlining the message as body of SOAP messages
5. Service S is provided with binding B at the following URI's (called ports)

WSDL Example

```
<?xml version="1.0">
   <definitions name="StockQuote>
   <types>
        <schema>
              definition of types in XML Schema .....
        </schema>
   </types>
   <message name="GetTradePriceInput">
        <part name="bid" type="xsd:string"/>
   </message>
   <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
              definition of an operation .....
        </operation>
   </portType>
   <binding name="StockQuoteSoapBinding">
        definition of a binding .....
   </binding>
   <service name="StockQuoteService">
        <port name="StockQuotePort">
              definition of a port .....
        </port>
   </service>
   </definitions>
```

1
2
3
4
5
6
7

WSDL Details

1	<definitions>	Root element
2	<types>	Data type definitions
3	<message>	Defines all messages
4	<portType>	Set of operations supported by a service
5	<binding>	Definitions of protocols etc.
6	<port>	Address for the binding
7	<service>	Aggregation of ports that belong together

C++ Example: TaskServer

```
int ns__getTask(std::string hostname,
                std::string &task);

int ns__finished(std::string ftask,
                 std::string &task);

int ns__error(std::string ftask,
              std::string &task);
```

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="WGTaskServer"
 targetNamespace="http://ludwig-sun2.unil.ch/~hstockin/WGTaskServer.wsdl"
 xmlns:tns="http://ludwig-sun2.unil.ch/~hstockin/WGTaskServer.wsdl"
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ns="urn:wgTaskServer"
 xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:WSDL="http://schemas.xmlsoap.org/wsdl/"
 xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>

 <schema targetNamespace="urn:wgTaskServer"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ns="urn:wgTaskServer"
  xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified">
  <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
 </schema>

</types>
```

1

2


```
<message name="getTaskRequest">
 <part name="hostname" type="xsd:string"/>
</message>

<message name="getTaskResponse">
 <part name="task" type="xsd:string"/>
</message>

<message name="finishedRequest">
 <part name="ftask" type="xsd:string"/>
</message>

<message name="finishedResponse">
 <part name="task" type="xsd:string"/>
</message>

<message name="errorRequest">
 <part name="ftask" type="xsd:string"/>
</message>

<message name="errorResponse">
 <part name="task" type="xsd:string"/>
</message>
```

3

```
int ns__getTask(std::string hostname,
                std::string &task);

int ns__finished(std::string ftask,
                 std::string &task);

int ns__error(std::string ftask,
              std::string &task);
```

4

```
<portType name="WGTaskServerPortType">
 <operation name="getTask">
  <documentation>Service definition of function ns__getTask
  </documentation>
  <input message="tns:getTaskRequest"/>
  <output message="tns:getTaskResponse"/>
 </operation>
 <operation name="finished">
  <documentation>Service definition of function ns__finished
  </documentation>
  <input message="tns:finishedRequest"/>
  <output message="tns:finishedResponse"/>
 </operation>
 <operation name="error">
  <documentation>Service definition of function ns__error
  </documentation>
  <input message="tns:errorRequest"/>
  <output message="tns:errorResponse"/>
 </operation>
</portType>
```


5

```
<binding name="WGTaskServer" type="tns:WGTaskServerPortType">
 <SOAP:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="getTask">
  <SOAP:operation style="rpc" soapAction=""/>
  <input>
     <SOAP:body use="encoded" namespace="urn:wgTaskServer" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/
  </input>
  <output>
    <SOAP:body use="encoded" namespace="urn:wgTaskServer" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
 </operation>
 <operation name="finished">
  <SOAP:operation style="rpc" soapAction=""/>
  <input>
    <SOAP:body use="encoded" namespace="urn:wgTaskServer" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
    <SOAP:body use="encoded" namespace="urn:wgTaskServer" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
 </operation>
 <operation name="error">
  <SOAP:operation style="rpc" soapAction=""/>
  <input>
     <SOAP:body use="encoded" namespace="urn:wgTaskServer" encodingStyle="http://sche
  </input>
  <output>
     <SOAP:body use="encoded" namespace="urn:wgTaskServer" encodingStyle="http://sche
  </output>
 </operation>
</binding>
```

6

7

```
<service name="WGTaskServer">
 <documentation>gSOAP 2.7.6e generated service definition
 </documentation>
 <port name="WGTaskServer" binding="tns:WGTaskServer">
  <SOAP:address location="http://localhost:8085"/>
 </port>
</service>

</definitions>
```

Note: this service uses RPC/encoded style (see next slides) and does not provide much documentation on the methods.

2

Data Types

- Typically, XML Schema is used for data types (`string`, `int`, etc.)

```
<types>
  <schema>
    <element name="PositionRequest">
      <complexType>
        <sequence>
          <element name="zipCode" type="string"/>
        <sequence>
      <complexType>
   <element>
  </schema>
<types>
```

PortTypes (1)

4

- WSDL supports 4 message patterns that an endpoint (=service provider!) can support for an operation
  - one-way: message is sent to service provider without expecting response
  - request-response: request is sent to service provider expecting response
  - solicit-response: provider sends a message and expects response
  - notification: message is sent by service provider

PortTypes (2)

4

- Message patterns are distinguished by the use of input/output elements
  - one way:

```
<definitions .... > <portType .... > *
     <operation name="doSomething">
        <input name="doSomething"? message="qname"/>
     </operation>
 </portType >
</definitions>
```

  - request/response:

```
<definitions .... >
 <portType .... > *
     <operation name="doSomething" parameterOrder="nmtokens">
        <input name="doSomething"? message="qname"/>
        <output name="doSomething"? message="qname"/>
        <fault name="doSomething" message="qname"/>*
     </operation>
 </portType >
</definitions>
```

5

Binding Style

```
<binding name="WGTaskServer"
 <SOAP:binding style="rpc" tr
<operation name="getTask">
 <SOAP:operation style="rpc"
 <input>
```

- Remote Procedure Call (RPC) Style
  - Messages are very similar to methods/functions in programming language
  - In the lecture, we mainly discuss RPC style
- Document Style
  - An alternative way to create messages
- Reading:

http://www-128.ibm.com/developerworks/webservices/library/ws-whichwsdl

5

```
public void myMethod(int x, float y);
```

```
<message name="myMethodRequest">
    <part name="x" type="xsd:int"/>
    <part name="y" type="xsd:float"/>
</message>
<message name="empty"/>

<portType name="PT">
    <operation name="myMethod">
        <input message="myMethodRequest"/>
        <output message="empty"/>
    </operation>
</portType>
```

RPC style

document style

```
<types>
    <schema>
        <element name="xElement" type="xsd:int"/>
        <element name="yElement" type="xsd:float"/>
    </schema>
</types>

<message name="myMethodRequest">
    <part name="x" element="xElement"/>
    <part name="y" element="yElement"/>
</message>
<message name="empty"/>

<portType name="PT">
    <operation name="myMethod">
        <input message="myMethodRequest"/>
        <output message="empty"/>
    </operation>
</portType>
```

5

Encoded vs. literal

```
public void myMethod(int x, float y);
```

```
<myMethod>
   <x type="int">5</x>
   <y type="float">5.0</y>
</myMethod>
```

```
<myMethod>
   <x>5</x>
   <y>5.0</y>
</myMethod>
```


5

Binding Style Summary

- The recommendation and most standard practise is to use
  - Document/literal wrapped style
- For details about “wrapped” read:

http://www-128.ibm.com/developerworks/webservices/library/ws-whichwsdl

Documentation in WSDL

- Add method and type documentation to your WSDL file

```
<element name="myMethod">
  <annotation>
   <documentation>Datatype elements can be documented in this way.
   If a wrapper element references other elements that are
   documented, documentation of the wrapper might not be
   necessary.
   </documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="x" type="int">
        <annotation>
         <documentation>Individual elements can also be
         documented.
         </documentation>
        </annotation>
      </element>
      <element name="y" type="float"/>
    </sequence>
  </complexType>
</element>
```


Web Service Interoperability (WS-I)

- SOAP is a well described standard
- However, different language implementations interpret it sometimes differently
  - Interoperability problems
- WS-I Organisation limits the usage of SOAP to allow for better interoperability (www.ws-i.org)
  - Examples
    - only “literal” style is allow (no “rpc”)
    - Restrictions on how to create arrays etc.

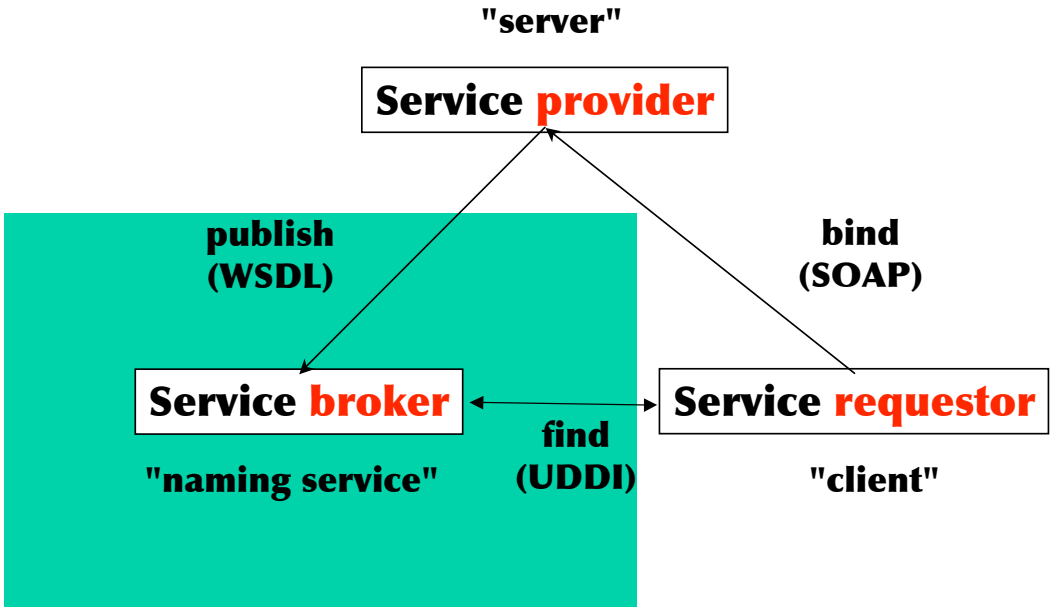
WS-I cont.

- Main interoperability standard is
  - WS-I Basic Profile (current versions 1.0 or 1.1)
  - http://www.ws-i.org/deliverables/workinggroup.aspx?wg=basicprofile
- Other standards also for
  - Security
  - Attachments to SOAP messages

WS-I Basic Profile Compliance

- WSDL file needs to be compliant
  - Test with WS-I Analyzer tool
  - Test with on-line tool (easier :-)
    - http://mindreef.net/tide/scopeit/start.do
- SOAP message exchange between client and server
  - Checks if SOAP/HTTP message is compliant with the standard
  - Test with WS-I Monitor

Reminder: Web Service Architecture



How to find/discover Web services?

- Who tells me where services are available?
- Is there a “telephone” book for services?
- Can I simply google them?

UDDI – Universal Description Discovery and Integration

- Standard for describing, publishing and finding Web services
  - Still evolving
  - Use XML-based description files for services
- Main components
  - White pages: basic contact information about an organization
  - Yellow pages: classification of organization based on industrial categorization
  - Green pages: technical description of services offered by registered organizations
- Access to UDDI Registry
  - Standard UDDI API (accessible via SOAP)
  - Web browser
- UDDI Data Structures (XML)
  - Business entity: general information + business services
  - Business services: business level description + binding templates
  - Binding templates: access point + tModel (service types)
  - tModel: abstract definition of a web service

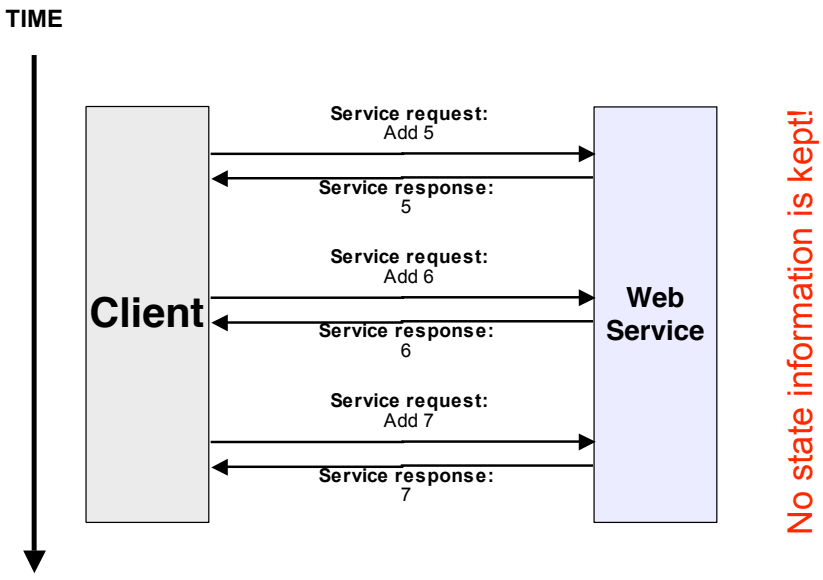
UDDI in Summary

- A UDDI registry is a Web service on its own
  - Knows about registered services
    - URL (location)
    - WSDL (service description)
- Most important features:
  - Publish information about your service
  - Search for other servers
- Example:
  - All Web services written by different teams in IIS course

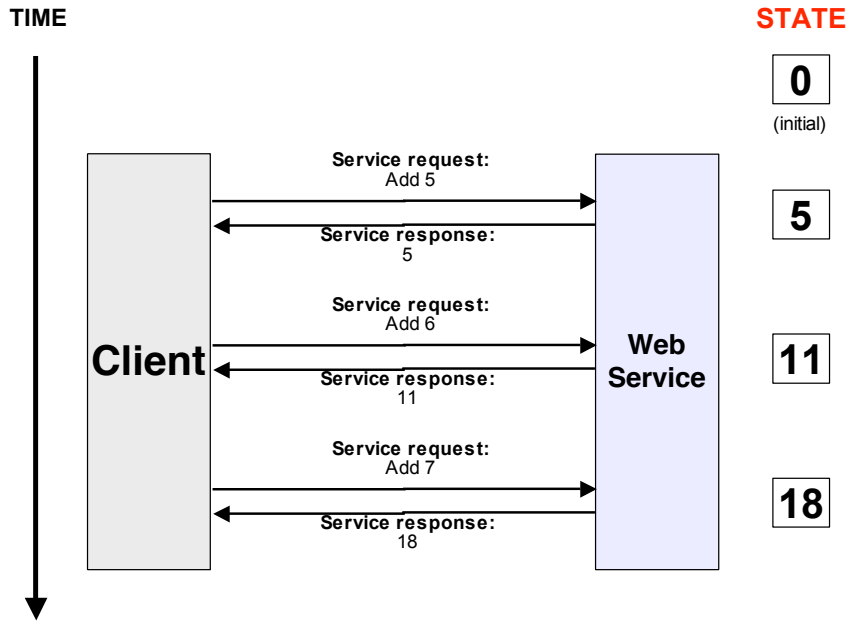
Other Web Service Standards

- WS-Security
- WS-Resource Framework (WS-RF)
  - Provides stateful Web Services
- WS-Agreement
  - Guaranteed service
- etc.

A stateless Web Service



A stateful Web Service



Web Services in Practice

- Serveral programming languages provide SOAP libraries:
  - Java: Axis, Axis2
  - C++: gSOAP, Axis2/C
  - Perl: soap::lite
  - Python: ZSI
  - etc.

Web Services in Practise: where to start?

- Start design your public interface
  - Can be in your favourite programming language
- Turn the public interface into a WSDL file
- Generate client and server stubs that you can use in your client and server code
- SOAP messages will be exchanged via the SOAP libraries (e.g. Axis2)
  - You do not need to worry about XML/SOAP/HTTP

REST - Representational State Transfer

- Originally, referred to as the "Architecture of the Web"
- Idea:
  - Do not need to use SOAP to access Web services
  - Rely on simple HTTP commands GET, PUT, POST etc.

REST Details

- The Web consists of resources
  - `http://www.example.com/book/1`
- A representation of a book is returned
  - `MyBook.html`
- A client changes state when transferring or accessing resources
- REST is an architectural style and not a standard
  - It *uses* standards such as HTTP, URL, XML, etc.

REST Example

`http://www.parts-depot.com/parts`

- The client receives:

```
<?xml version="1.0"?>
<p:Parts xmlns:p="http://www.parts-depot.com"
         xmlns:xlink="http://www.w3.org/1999/xlink">
      <Part id="00345" xlink:href="http://www.parts-depot.com/parts/00345"/>
      <Part id="00346" xlink:href="http://www.parts-depot.com/parts/00346"/>
      <Part id="00347" xlink:href="http://www.parts-depot.com/parts/00347"/>
      <Part id="00348" xlink:href="http://www.parts-depot.com/parts/00348"/>
</p:Parts>
```

- Get detailed part `http://www.parts-depot.com/parts/00345`

```
<?xml version="1.0"?>
<p:Part xmlns:p="http://www.parts-depot.com"
        xmlns:xlink="http://www.w3.org/1999/xlink">
      <Part-ID>00345</Part-ID>
      <Name>Widget-A</Name>
      <Description>This part is used within the frap assembly</Description>
      <Specification xlink:href="http://www.parts-depot.com/parts/00345/specification"/>
      <UnitCost currency="USD">0.10</UnitCost>
      <Quantity>10</Quantity>
</p:Part>
```

http://www.xfront.com/REST-Web-Services.html

REST versus SOAP

- REST is light weight and rather easy to integrate with any programming language
- SOAP/WSDL requires clearly defined interfaces and data types:
  - Code can be used to integrate in workflows
  - Rich metadata

Conclusion

- Web services are a very popular way to build client-server applications
- SOAP implementations of different languages take care of the details
  - WSDL can be auto-generated from
  - Language client can be auto-generated from WSDL
- SOAP is a language independent protocol
- Good overview:
  - http://www.w3schools.com/soap/
- REST is also very popular and simpler

References

- Standard documents
  - http://www.w3.org/2002/ws/
  - http://www.w3.org/TR/2002/CR-soap12-part0-20021219/ (SOAP primer)
  - http://www.w3.org/TR/SOAP/
  - http://www.w3.org/TR/wsdl
  - http://www.uddi.org
- http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm