

# CmpE 494: Service-Oriented Architectures and Web Services

Pinar Yolum

`pinar.yolum@boun.edu.tr`

Department of Computer Engineering  
Boğaziçi University

# Part IV: SOAP

# Simple Object Access Protocol

- XML-based protocol for exchanging structured messages
- Independent of platform or programming language
- Can use various transport protocols; commonly HTTP
- Can be extended easily for different needs
  - Basis for other Web service activities (security, trust)
  - Layered design

# SOAP Messages

- Is contained in an envelope
- Consists of a header (optional) and a body (mandatory)
- Destined from a SOAP sender to a SOAP receiver along a message path
- There may be intermediaries in between
- Intermediaries may inspect and modify the header
- Messages may be transferred over any network protocol

# Internals of SOAP Message

- Body contains main matter
- Header contains control information
  - How should the message be processed by the SOAP receiver?
  - How should the message be handled by intermediaries?
- Header consists of header blocks

# Example SOAP Message (Header)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <n:name>Yolum</n:name>
    </n:passenger>
  </env:Header>
```

# Example SOAP Message (Body)

```
<env:Body>
  <p:itinerary
    xmlns:p="http://travelcompany.example.org/reservation/travel">
    <p:departure>
      <p:departing>New York</p:departing>
      <p:arriving>Los Angeles</p:arriving>
      <p:departureDate>2001-12-14</p:departureDate>
      <p:departureTime>late afternoon</p:departureTime>
      <p:seatPreference>aisle</p:seatPreference>
    </p:departure>
  </p:itinerary>
  <q:lodging
    xmlns:q="http://travelcompany.example.org/reservation/hotels">
    <q:preference>none</q:preference>
  </q:lodging>
</env:Body>
</env:Envelope>
```

# Transfer of a SOAP Message

- env:role
  - env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
  - env:role="http://www.w3.org/2003/05/soap-envelope/role/none"
  - env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
- env:mustUnderstand="true" :Receiving node must process the header according to the specification
- env:relay="true" :Denotes that the header must be forwarded if not processed (rather than raising a fault)

# Message Pattern

- The message is finally received by whoever claims to be the ultimateReceiver.
  - What would be a problem?
- SOAP Request-Response
  - Response can change the body as well as the header
- SOAP Response
- SOAP requests may involve remote procedure calls (RPCs)

# Response SOAP Message (Header)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateAndTime>2001-11-29T13:35:00.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <n:name>Yolum</n:name>
    </n:passenger>
  </env:Header>
```

# Response SOAP Message (Body)

```
<env:Body>
  <p:itineraryClarification
    xmlns:p="http://travelcompany.example.org/reservation/travel">
    <p:departure>
      <p:departing>
        <p:airportChoices>JFK LGA EWR</p:airportChoices>
      </p:departing>
    </p:departure>
  </p:itineraryClarification>
</env:Body>
</env:Envelope>
```

# SOAP RPC

- The address of the target SOAP node
- Method to be invoked and its arguments
- Separation of identification data from control data
- Message exchange pattern

# SOAP RPC Example

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/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:chargeReservation
      env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
      xmlns:m="http://travelcompany.example.org/" >
      <m:reservation xmlns:m="http://travelcompany.example.org/reservation" >
        <m:code>FT35ZBQ</m:code>
      </m:reservation>
    </m:chargeReservation>
  </env:Body>
</env:Envelope>
```

# SOAP RPC Example

```
<o:creditCard xmlns:o="http://mycompany.example.com/financial">  
  <n:name xmlns:n="http://mycompany.example.com/employees">  
    Yolum  
  </n:name>  
  <o:number>123456789099999</o:number>  
  <o:expiration>2005-02</o:expiration>  
</o:creditCard>  
</m:chargeReservation>  
</env:Body>  
</env:Envelope>
```

# SOAP RPC

- Requesting RPC
  - <chargeReservation> (name of the method to be invoked)
  - Children (<reservation>, <creditCard>) are the parameters to the method
  - <transaction> denotes that the method invocation is part of a transaction
- Responding to a RPC
  - Responses contain the method name and append “Response”
  - Children of the response contain the returned values

# SOAP RPC Return Example

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/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/2003/05/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>
```

# Designated Return Value

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/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/2003/05/soap-encoding"
      xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
      xmlns:m="http://travelcompany.example.org/">
      <rpc:result>m:status</rpc:result>
      <m:status>confirmed</m:status>
      <m:code>FT35ZBQ</m:code>
      <m:viewAt>
        http://travelcompany.example.org/reservations?code=FT35ZBQ
      </m:viewAt>
    </m:chargeReservationResponse>
  </env:Body>
```

# SOAP Fault Messages

- Sent to the originator when there is a fault
- Denoted by env:Fault
  - env:Code
    - Contains a mandatory env:Value
    - Contains an optional env:SubValue
  - env:Reason—meant for human understanding
  - env:Detail—application-specific information (optional)
  - env:Role—the role that generated the fault (optional)
- Separation of identification data from control data
- Message exchange pattern

# Example SOAP Fault Message (Part 1)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:rpc='http://www.w3.org/2003/05/soap-rpc'>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value>rpc:BadArguments</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US">Processing error</env:Text>
        <env:Text xml:lang="cs">Chyba zpracovn</env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

# Example SOAP Fault Message (Part 2)

```
<env:Detail>
  <e:myFaultDetails
    xmlns:e="http://travelcompany.example.org/faults">
    <e:message>Name does not match card number</e:message>
    <e:errorCode>999</e:errorCode>
  </e:myFaultDetails>
</env:Detail>
</env:Fault>
</env:Body>
</env:Envelope>
```

# Example SOAP Fault Message

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:NotUnderstood qname="t:transaction"
      xmlns:t="http://thirdparty.example.org/transaction"/>
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:MustUnderstand</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US">Header not understood</env:Text>
        <env:Text xml:lang="fr">En-tte non compris</env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

# Protocol Bindings

- Specification of how messages will be transported among SOAP nodes
  - Serialize the message content; typically by XML serialization
  - Support additional features such as correlating requests with responses
- Must support SOAP message patterns (such as Request/Response)
- SOAP specification allows Web methods such as GET and POST
- Current available binding is HTTP with GET and POST

# SOAP HTTP GET

- Send an HTTP GET message to receive a SOAP message

```
GET /travelcompany.example.org/reservations?code=FT35ZBQ HTTP/1.1
```

```
Host: travelcompany.example.org
```

```
Accept: text/html;q=0.5, application/soap+xml
```

- Accept denotes how the response is requested
- The response is a SOAP message with the following prefix:

```
HTTP/1.1 200 OK
```

```
Content-Type: application/soap+xml; charset="utf-8"
```

```
Content-Length: nnnn
```

# SOAP HTTP POST

- Send the request SOAP message inside an HTTP POST message

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

- The response SOAP message is also encoded in HTTP

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

- 2xx codes stand for success; 500 stands for Internal Server Error, etc.

# SOAP Intermediaries

- Forwarding intermediary
  - Processes header blocks that are targeted for next roles
  - Does not change the header
- Active intermediary
  - Processes header blocks that are targeted for next roles
  - May modify the existing header blocks or add new blocks