Web Service Description Language (WSDL)

02/09/15

WSDL

- Web Service Description Language (WSDL)
- WSDL describes how to interact with a particular service
  - i.e. defines the service contract
- XML-based ”Web IDL” (cf. CORBA IDLs)
- Contains descriptions of three main parts, each of them can also be given as a separate document (and later combined to form a complete WSDL description)
  - the data, typically using XML schemas
  - operations to be performed on the data
    - for the message receiver to know how to process the message
  - binding to a protocol or transport
    - for the message sender to know how to send the message
- For more info, see: W3C “Web services” activity group
  - WSDL version 2.0: W3C Recommendation (June 2007)

02/09/15

WSDL-kieli ja erityisesti sen versio 2.0 on hyväksytty kesäkuussa 2007 W3C:n viralliseksi suositukseksi. Yleisesti käytössä oleva versio on edelleen WSDL 1.1., jota esimerkiksi Web-palvelujen yhteentoimivuuskysymyksiin keskittyvä WS-I organisaatio käyttää profiilimäärityksissään (Basic Profile v. 1.0, v. 1.1 ja myös v. 1.2). Profiilimääritys käytännössä antaa ohjeistusta siitä, miten ”Web-palvelustandardeja” tulisi käyttää, jotta yhteensopivuus olisi mahdollisimman todennäköistä. WS-I organisaation suosituksia seurataan huolella ja suurin osa suurimmista Web-palvelutyökalujen toteuttajista huomioi ne työkalujensa uusimmissa versioissa. Tällä kurssilla WSDL-kieli käydään läpi version 2.0 pohjalta, mutta koska myös versio 1.1. on edelleen melko laajalti käytössä, käydään pääerot näiden versioiden välillä myös läpi.

WSDL-kielen on sanottu olevan Web-palveluille sama kuin IDL on CORBAlle. XML-pohjainen WSDL on käytetyin tapa etsiä ja paikallistaa Web-palveluita. Se onkin näin ollen avain sovellusten väliseen kommunikointiin Web-palvelukonseptissa. WSDL-kuvaukset ovat riippumattomia viestinvälitysmekanismista, mutta SOAP-sidonta on yleisimmin käytetty.

Yksi hyvä tapa tutustua WSDL-kieleen ja oppia sitä on käydä läpi esimerkkejä.

WSDL structure

```
<?xml version="1.0" encoding="utf-8"?>
<description
    xmlns="http://www.w3.org/ns/wsdl"
    … >
  <types>…</types>
  <interface>…</interface>
  <binding>…</binding>
  <service>…</service>
</description>
```

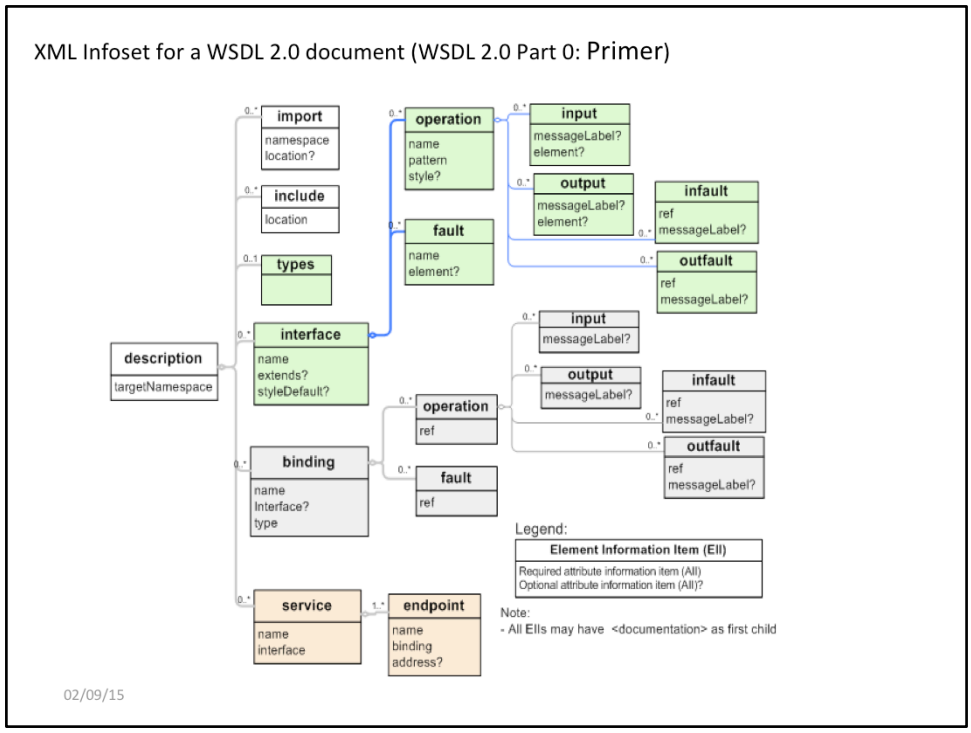
Main parts of a WSDL 2.0 document

- The root element of a WSDL 2.0 document is **`description`**.
- The four main parts of a WSDL 2.0 document are structured under the following subelements of description:
  - **`types`**: describes the kinds of messages that the service will send and receive
  - **`interface`**: describes what abstract functionality the Web service provides
  - **`binding`**: describes how to access the service
  - **`service`**: describes where to access the service

02/09/15

WSDL 2.0 dokumentti koostuu neljästä pääosasta:
-*types*: kuvaa millaisia viestejä palvelu lähettää ja vastaanottaa
-*interface*: kuvaa minkä abstraktin toiminnallisuuden Web-palvelu tarjoaa
-*binding*: kuvaa miten palveluun voidaan ottaa yhteyttä
-*service*: kuvaa missä palvelu sijaitsee

Nämä neljä osiota kuvataan omina alirakenteinaan juurielementin *description* alla. Verrattuna versioon 1.1, osa elementtien nimistä on muuttunut. Esimerkiksi juurielementti versiossa 1.1 oli nimeltään *definitions* ja rajapinnan kuvaava elementti (interface) oli nimeltään *portType.* Myös muutamia muita elementtien uudelleennimeämisiä on tehty. Nämä muutokset ovat olleet hyviä, sillä uudet nimet kuvaavat niiden tarkoitusta paremmin.



Kalvolla esitetty WSDL 2.0 dokumentin informaatiosisältö. Kuva on W3C:n dokumentista WSDL 2.0 Part0: Primer. Tämä informaatiomalli kuvaa WSDL 2.0 dokumentin vaaditun rakenteen. WSDL 2.0 –kieli sisältää myös useita semanttisia rajoitteita rakennetta koskevien vaatimusten lisäksi. Näiden rajoitteiden kuvaamiseksi ja toisaalta WSDL 2.0 dokumentin merkityksen määräämiseksi WSDL 2.0 spesifikaatio määrittelee myös nk. *komponenttimallin* (component model). Kyseinen komponenttimalli vastaa informaatiomallia ja onkin esitetty abstraktina erillisenä kerroksen ko. informaatiomallille.

<types>

- `types` element defines the data types used in the service interface
- Types are typically given using XML Schema
  - also other schema definition languages can be used, e.g RELAX NG and Schematron
- Extensible and replaceable

Elementti *types* määrittelee käytettävät tietotyypit. Tämä osio voidaan antaa myös omana dokumenttinaan. Koska tyyppimääritykset annetaan omana osionaan, voidaan se helposti korvata vaihtoehtoisella tyyppimäärittelyllä.

<types>

```
<types>
  <xs:schema targetNamespace="http://greath.example.com/2004/schemas/
resSvc">
    <xs:element name="checkAvailability" type="tCheckAvailability"/>
    <xs:complexType name="tCheckAvailability">
      <xs:sequence>
        <xs:element name="checkInDate" type="xs:date"/>
        <xs:element name="checkOutDate" type="xs:date"/>
        <xs:element name="roomType" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:element name="checkAvailabilityResponse" type="xs:double"/>
    <xs:element name="invalidDataError" type="xs:string"/>
  </xs:schema>
</types>
```

Esimerkki. WSDL Version 2.0 Part 0: Primer
http://www.w3.org/TR/wsdl20-primer/

<interface>

- `interface` element defines the abstract interface of the service
- Interface consists of `operation` elements

Elementti *interface* määrittelee abstraktin rajapinnan Web-palvelulle joukkona operaatioita, jotka on puolestaan annettu *operation*-elementtien avulla.

Elementti *operation* kuvaa yksinkertaisen interaktion asiakkaan ja palvelun välillä. Se määrittelee myös palvelun lähettämien ja vastaanottamien viestien tyypit.

<interface> <operation>

- `operation` element represents a simple interaction between the client and the service
- Specifies the types of messages that the service can send or receive as part of that operation
- Specifies a Message Exchange Pattern
- contains
  - A required name attribute, which must be unique within the interface.
  - A required pattern attribute whose value must be an absolute URI that identifies the desired MEP for the operation.
  - An optional style attribute whose value is a list of absolute URIs (RPC style, IRI style, Multipart style)

interface-elementin lapsielementti *operation* kuvaa yksinkertaisen interaktion asiakkaan ja palvelun välillä. Se määrittelee myös palvelun lähettämien ja vastaanottamien viestien tyypit.

Message Exchange Patterns

- WSDL 2.0 Adjuncts defines eight Message Exchange Patterns (MEP)
  - In-bound MEPs, for which the service receives the first message in the exchange
    - In-Only
    - Robust In-Only
    - In-Out
    - In-Optional-Out
  - Out-bound MEPs, for which the service sends out the first message in the exchange
    - Out-Only
    - Robust Out-Only
    - Out-In
    - Out-Optinal-In

Elementti *operation* assosioi käytetyt viestimallit (message exchange patterns). WSDL 2.0 tukee seuraavia viestimalleja:

1) MEPit, jotka alkavat palvelun vastaanottamasta viestistä
   1) In-Only: koostuu täsmälleen yhdestä viestistä, jonka palvelu vastaanottaa. Virheviestiä ei generoida.
   2) Robust In-Only: koostuu täsmälleen yhdestä viestistä, jonka palvelu vastaanottaa. Virheviesti voidaan generoida ja se tulee lähettää viestin lähettäjälle.
   3) In-Out: koostuu täsmälleen kahdesta viestistä: vastaanotetusta ja lähetetystä viestistä. Vaste (lähetetty vastaus) voidaan korvata generoidulla virheviestillä.
   4) In-Optional-Out: koostuu yhdestä tai kahdesta viestistä määrätyssä järjestyksessä: vastaanotettu pyyntö ja sen jälkeen optionaalinen lähetetty vaste. Kumpi tahansa MEPin viesti voi generoida virheen. Virheviestin suunnan tulee olla vastakkainen ko. viestille.

2) MEPit, jotka alkavat lähetetystä viestistä
   1) Out-Only: koostuu täsmälleen yhdestä palvelun lähettämästä viestistä. Virheviestiä ei generoida.
   2) Robust Out-Only: koostuu täsmälleen yhdestä palvelun lähettämästä viestistä. Virheviesti voidaan generoida ja se lähetetään interaktion aloittajalle.
   3) Out-In: koostuu täsmälleen kahdesta viestistä: pyyntö ja vaste. Nämä viestit tulee olla tässä järjestyksessä. Vaste voidaan korvata generoidulla virheviestillä.
   4) Out-Optional-In: koostuu yhdestä tai kahdesta viestistä: lähetetty pyyntö ja optionaalinen vaste. Kumpi tahansa MEPin viesti voi generoida virheen. Virheviestin suunnan tulee olla vastakkainen ko. viestille.

<interface>

```
<interface name="reservationInterface">
  <fault name="invalidDataFault“
         element="ghns:invalidDataError"/>
  <operation name="opCheckAvailability“
      pattern="http://www.w3.org/ns/wsdl/in-out"
      style="http://www.w3.org/ns/wsdl/style/iri"
      wsdlx:safe="true">
    <input messageLabel="In"
           element="ghns:checkAvailability"/>
    <output messageLabel="Out"
            element="ghns:checkAvailabilityResponse"/>
    <outfault ref="tns:invalidDataFault“
              messageLabel="Out"/>
  </operation>
</interface>
```

<binding>

- `binding` element specifies *how* the messages can be exchanged
  - i.e. how to bind interface to a particular transport (e.g., SOAP)
- Must supply details on concrete message formats and transmission protocols for every operation and fault in the interface.
- Attributes
  - `name` (required): defines (uniquely among all bindings in the WSDL 2.0 target namespace) a name for this binding.
  - `interface` (optional): the name of the interface whose message format and transmission protocols are specified
  - `type` (required): specifies what kind of concrete message format to use
  - `protocol`: WSDL 2.0 SOAP binding extension specific attiribute that specifies the underlying transmission protocol that should be used

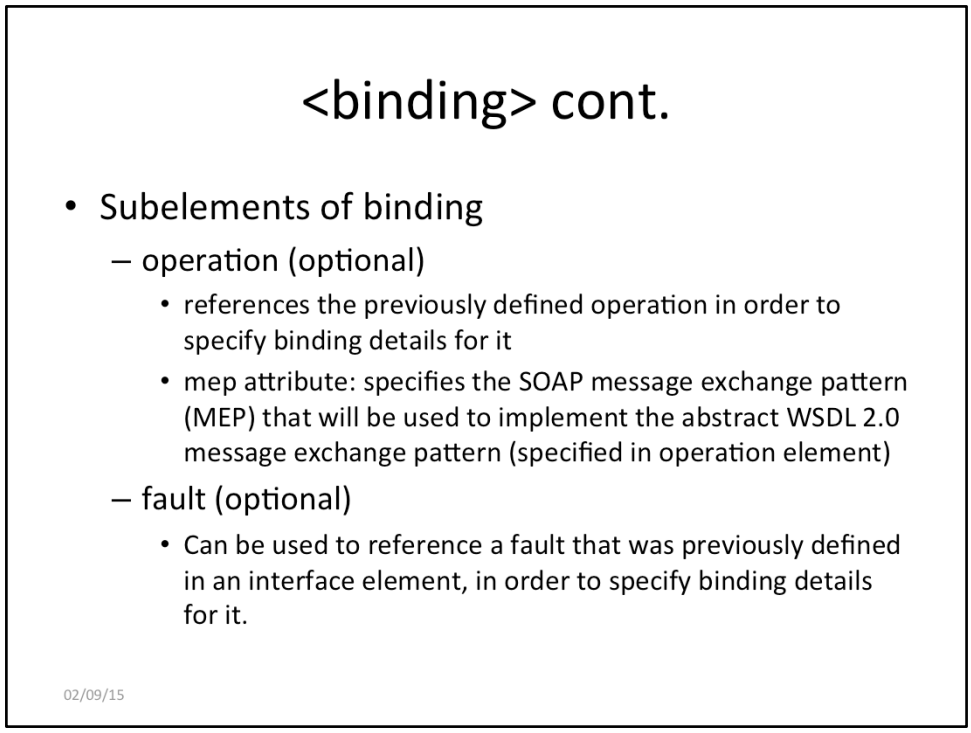
02/09/15

WSDL-dokumentissa on operaatioiden kuvausten lisäksi määritelty niiden sitominen viestinvälitysmekanismeihin. Tämä määritellään *binding*-elementin avulla. WSDL 2.0 sisältää SOAP-laajennoksen, jossa kuvataan miten tämä sitominen tulee tehdä käytettäessä SOAPia kommunikointitapana. WSDL ei itsessään oleta SOAPin käyttöä. Muitakin viestinvälitysmekanismeja voidaan toki käyttää. Koska SOAP on kuitenkin ehkäpä se yleisin tapa ja koska se on erityisesti standardi tapa Web-palvelukonseptissa, on WSDL-spesifikaatioon tehty tämä laajennos.

binding-elementillä on kolme attribuuttia. Pakollisella *name*-attribuutilla nimetään annettu sidonta (binding). Nimen tulee olla yksikäsitteinen kyseisessä WSDL-kuvauksen kohdenimiavaruudessa. Tähän nimeen viitataan myöhemmin *endpoint*-elementistä. Optionaalisella attribuutilla *interface* puolestaan määritellään rajapinta (interface), jonka viestiformaatti ja kuljetusprotokolla tässä *binding*-elementissä tullaan määrittämään. Pakollisella attribuutilla *type* puolestaan määritetään mitä konkreettista viestiformaattia käytetään. Esimerkiksi käytettäessä SOAPin versiota 1.2, *type*-attribuutin arvoksi annetaan ” http://www.w3.org/ns/wsdl/soap”

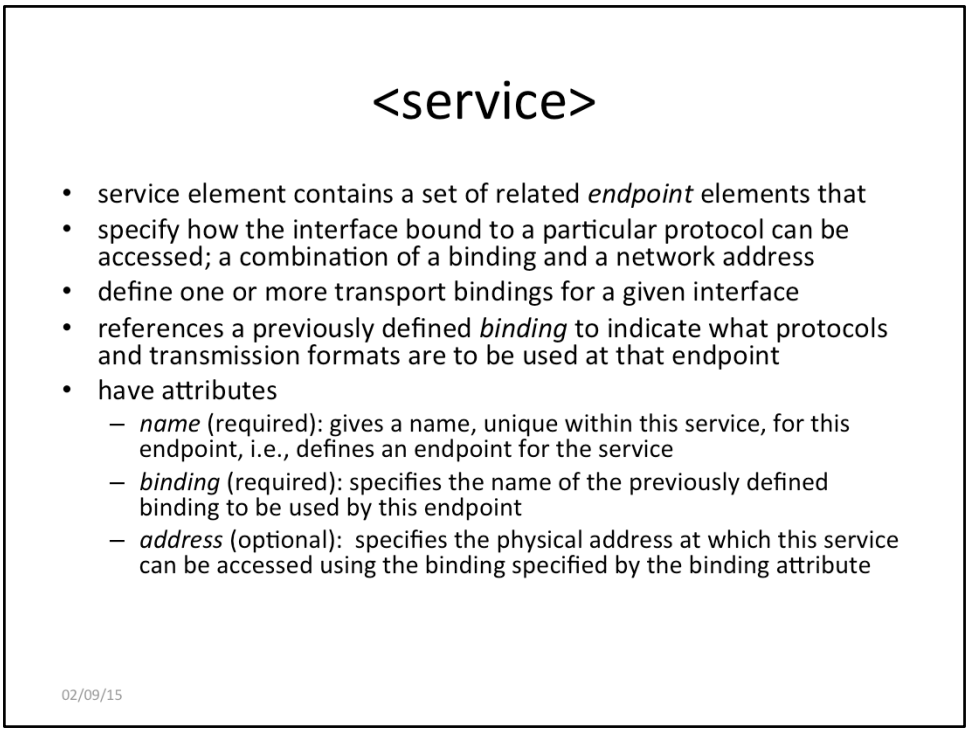
<binding>

```
<description
   …
   xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
   xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
   xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
…
<binding name="reservationSOAPBinding"
        interface="tns:reservationInterface"
        type="http://www.w3.org/ns/wsdl/soap"
        wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">

   <operation ref="tns:opCheckAvailability"
     wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

   <fault ref="tns:invalidDataFault"
     wsoap:code="soap:Sender"/>

 </binding>
…
```



WSDL-dokumentissa on operaatioiden kuvausten lisäksi määritelty niiden sitominen viestinvälitysmekanismeihin. Tämä määritellään *binding*-elementin avulla. WSDL 2.0 sisältää SOAP-laajennoksen, jossa kuvataan miten tämä sitominen tulee tehdä käytettäessä SOAPia kommunikointitapana. WSDL ei itsessään oleta SOAPin käyttöä. Muitakin viestinvälitysmekanismeja voidaan toki käyttää. Koska SOAP on kuitenkin ehkäpä se yleisin tapa ja koska se on erityisesti standardi tapa Web-palvelukonseptissa, on WSDL-spesifikaatioon tehty tämä laajennos.

binding-elementillä on kolme attribuuttia. Pakollisella *name*-attribuutilla nimetään annettu sidonta (binding). Nimen tulee olla yksikäsitteinen kyseisessä WSDL-kuvauksen kohdenimiavaruudessa. Tähän nimeen viitataan myöhemmin *endpoint*-elementistä. Optionaalisella attribuutilla *interface* puolestaan määritellään rajapinta (interface), jonka viestiformaatti ja kuljetusprotokolla tässä *binding*-elementissä tullaan määrittämään. Pakollisella attribuutilla *type* puolestaan määritetään mitä konkreettista viestiformaattia käytetään. Esimerkiksi käytettäessä SOAPin versiota 1.2, *type*-attribuutin arvoksi annetaan ” http:// www.w3.org/ns/wsdl/soap”



Sidonta (binding) kuvaa *miten* viestit välitetään. Tämän jälkeen WSDL-kuvauksessa tulee vielä määrittää *missä* palvelut sijaitsevat, jotta niihin voitaisiin ottaa yhteyttä. Tämä tehdään *service*-elementin avulla. *service*-elementti sisältää *endpoint*-alielementtejä. Jokainen *endpoint*-elementti määrittää osoitteen tietylle viestinvälitysmekanismiin sidotulle rajapinnalle (interface). Näitä *endpoint*-elementtejä voi olla useita, koska palvelulle voidaan määrittää useita yhteydenottotapoja. Oletetaan esimerkiksi, että ”checkAvailability” palvelun *interface* on toteutettu kahdella eri tavalla (kaksi eri kuljetusprotokollaa): SOAP/HTTP ja SOAP/SMTP. Tällöin yksi *service* elementti voisi sisältää kontaktipisteen (endpoint), joka kuvaa URL:n SOAP/HTTP:lle, sekä kontaktipisteen, joka kuvaa sähköpostiosoitteen SOAP/SMTP:lle. Vastoin kuin WSDL:n versiossa 1.1, versiossa 2.0 *service*-elementillä voi olla vain yksi rajapinta (interface), johon viitataan attribuutilla *interface.*

<service> cont.

- Service element is used, e.g., for
  - grouping those endpoints related to the same service interface that are expressed by different protocols (bindings)
- a service is only permitted to have one interface
  - Referenced by attribute *interface*
- Attributes
  - *name* (required): defines (uniquely among all bindings in the WSDL 2.0 target namespace) a name for this service
  - *interface* (required): specifies the name of the previously defined interface that these service endpoints will support

<service>

- `service` element defines *where* the service can be accessed
  - Contains `endpoint` elements
- Attributes
  - *name* (required): defines (uniquely among all bindings in the WSDL 2.0 target namespace) a name for this service
  - *interface* (required): specifies the name of the previously defined interface that these service endpoints will support

<service> <endpoint>

- `endpoint` elements specify how the interface bound to a particular protocol can be accessed
  - a combination of a binding and a network address
- define one or more transport bindings for a given interface
- references a previously defined *binding* to indicate what protocols and transmission formats are to be used at that endpoint
- have attributes
  - *name* (required): gives a name, unique within this service, for this endpoint, i.e., defines an endpoint for the service
  - *binding* (required): specifies the name of the previously defined binding to be used by this endpoint
  - *address* (optional): specifies the physical address at which this service can be accessed using the binding specified by the binding attribute

02/09/15

Sidonta (binding) kuvaa *miten* viestit välitetään. Tämän jälkeen WSDL-kuvauksessa tulee vielä määrittää *missä* palvelut sijaitsevat, jotta niihin voitaisiin ottaa yhteyttä. Tämä tehdään *service*-elementin avulla.

service-elementti sisältää *endpoint*-alielementtejä. Jokainen *endpoint*-elementti määrittää osoitteen tietylle viestinvälitysmekanismiin sidotulle rajapinnalle (interface). Näitä *endpoint*-elementtejä voi olla useita, koska palvelulle voidaan määrittää useita yhteydenottotapoja. Oletetaan esimerkiksi, että ”checkAvailability” palvelun *interface* on toteutettu kahdella eri tavalla (kaksi eri kuljetusprotokollaa): SOAP/HTTP ja SOAP/SMTP. Tällöin yksi *service* elementti voisi sisältää kontaktipisteen (endpoint), joka kuvaa URL:n SOAP/HTTP:lle, sekä kontaktipisteen, joka kuvaa sähköpostiosoitteen SOAP/SMTP:lle. Vastoin kuin WSDL:n versiossa 1.1, versiossa 2.0 *service*-elementillä voi olla vain yksi rajapinta (interface), johon viitataan attribuutilla *interface.*

<service>

```
<service name="reservationService"
  interface="tns:reservationInterface">
 <endpoint name="reservationEndpoint"
   binding="tns:reservationSOAPBinding“
   address="http://greath.example.com/2004/reservation"/>
</service>
```

Importing and including

- A WSDL definition can consist of multiple XML documents
- Importing and including mechanisms can be used to combine the different parts into one WSDL definition
- WSDL 2.0
  - import: importing WSDL definitions with a different namespace
    - required namespace attribute that specifies the other namespace
    - optional location attribute that gives the processor a hint where to find the description of the other namespace
  - include: importing WSDL definitions with the same namespace
- WSDL 1.1
  - import: allows importing WSDL definitions from separate files, the namespaces can be different or the same

02/09/15

WSDL-määritykset voivat olla hyvinkin pitkiä. Usein onkin tarkoituksenmukaista jakaa tällainen määritys useampaan erilliseen osaan (ja erillisiin tiedostoihin). Esimerkiksi tyyppimääritykset, jotka annetaan *types*-osiossa, voidaan määritellä erillisessä tiedostossa, joka voidaan sitten sisällyttää useampaankin WSDL-kuvaukseen. Tämä mekanismi tukee näin ollen myös uudelleenkäyttöä.

WSDL:n versiot 2.0 ja 1.1 poikkeavat hieman myös tämän WSDL-dokumentin osion linkittämisen suhteen. WSDL 1.1 käyttää elementtiä *import,* jonka avulla päädokumenttiin voidaan sisällyttää WSDL-määrityksiä muista tiedostoista, riippumatta siitä kuuluvatko niiden elementit samaan tai eri nimiavaruuteen. WSDL 2.0 puolestaan käyttää elementtejä *import* ja *include,* joista edellistä käytetään sisällyttämään eri nimiavaruuteen kuuluvat määritykset ja jälkimmäistä taas samaan nimiavaruuteen kuuluvat määritykset.

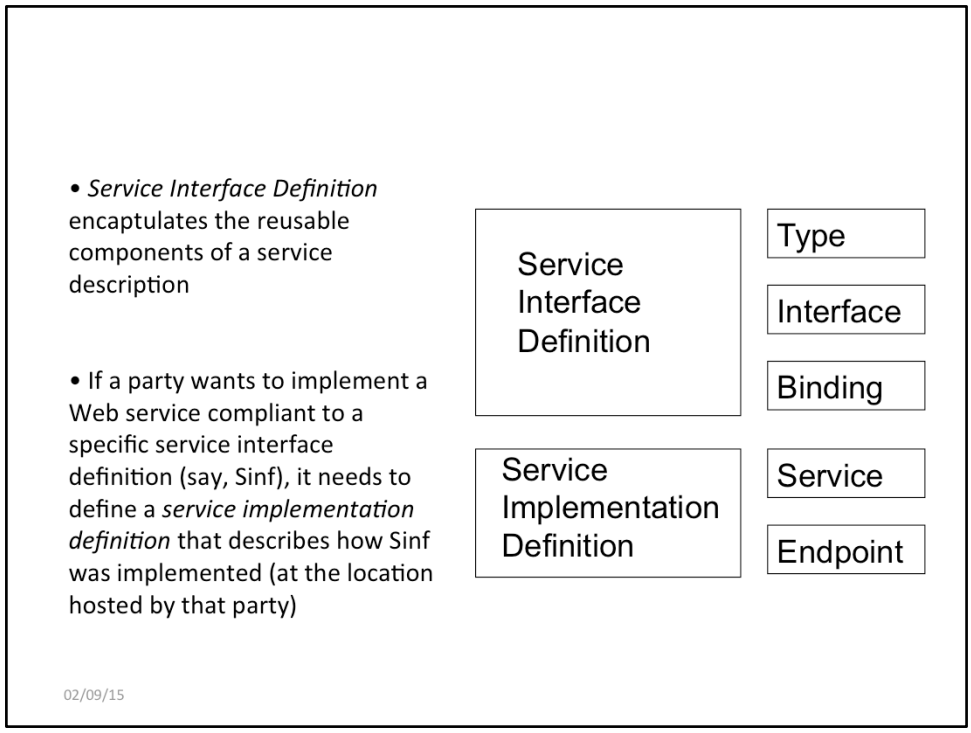
```
<description targetNamespace="http://finance.example.com/CreditCards/wsdl">
  <documentation>This document describes standard faults for use by Web
services that process credit cards.</documentation>
  <types>
    <xs:import namespace="http://finance.example.com/CreditCardFaults/xsd"
schemaLocation="credit-card-faults.xsd"/>
  </types>
  <interface name="creditCardFaults">
    <fault name="cancelledCreditCard" element="cc:CancelledCreditCard">
      <documentation>Thrown when the credit card has been cancelled.</
documentation>
    </fault>
    <fault name="expiredCreditCard" element="cc:ExpiredCreditCard">
      <documentation>Thrown when the credit card has expired.</
documentation>
    </fault>
    <fault name="invalidCreditCardNumber"
element="cc:InvalidCreditCardNumber">
      <documentation>Thrown when the credit card number is invalid?.</
documentation>
    </fault>
    <fault name="invalidExpirationDate" element="cc:InvalidExpirationDate">
      <documentation>Thrown when the expiration date is invalid.</
documentation>
    </fault>
  </interface>
</description>
```

Esimerkki (WSDL 2.0 Primer): määritellään rajapinta creditCardFaults, joka sisältää neljä virhekoodia: cancelledCreditCard, expiredCreditCard, invalidCreditCardNumber ja invalidExpirationDate. Nämä komponentit kuuluvat nimiavaruuteen http://finance.example.com/CreditCards/wsdl.

```
<description targetNamespace="http://greath.example.com/2004/wsdl/resSvc">
  <documentation>Description: The definition of the reservation Web service of
 GreatH hotel. Author: Joe Somebody Date: 05/17/2004</documentation>
  <import namespace="http://finance.example.com/CreditCards/wsdl"
 location="credit-card-faults.wsdl"/>
 ...
  <interface name="reservation" extends="cc:creditCardFaults">
    …
    <operation name="makeReservation" pattern="http://www.w3.org/ns/wsdl/in-
 out">
      <input messageLabel="In" element="ghns:makeReservation"/>
      <output messageLabel="Out" element="ghns:makeReservationResponse"/>
      <outfault ref="invalidDataFault" messageLabel="Out"/>
      <outfault ref="cc:cancelledCreditCard" messageLabel="Out"/>
      <outfault ref="cc:expiredCreditCard" messageLabel="Out"/>
      <outfault ref="cc:invalidCreditCardNumber" messageLabel="Out"/>
      <outfault ref="cc:invalidExpirationDate" messageLabel="Out"/>
    </operation>
  </interface>
</description>
```

02/09/15

2) Edellä määriteltyjä virhekoodeja käytetään GreatH-hotellinvarauspalvelussa, joka on määritelty alla. Rajapinta *reservation* perii laajentamalla *creditCardFaults*-rajapinnan toisesta nimiavaruudesta, jotta virhekoodit olisivat käytettävissä *reservation*-rajapinnassa. Lopuksi *makeReservation*-operaatio viittaa virhe-elementtiin *outfaults.*



WSDL-kuvaus voidaan ajatella kaksiosaiseksi: palvelun rajapinnan määritys (Service Interface Definition) ja toteutuksen määritys (Service Implementation Definition). Palvelun rajapinnan määritys on (tyypilliseen tapaan) uudelleenkäytettävä siinä mielessä, että sille voidaan antaa useita toteutuksia. Palvelun rajapinnan määritys on Web-palvelun abstrakti kuvaus ja sitä käytetään kuvaamaan tietyntyyppinen palvelu. Palvelun rajapinnan määritys koostuu elementeistä *types, interface,* ja *binding*. Palvelun toteutuksen määrittely puolestaan kuvaa palvelun instanssit. Toisin sanoen, palvelun toteutuksen määritys kertoo miten tietty rajapinta on toteutettu kyseisessä kohteessa.

Palvelun rajapinnan määrityksessä voidaan lisäksi käyttää elementtiä *import*, jonka avulla rajapinnan määritys voidaan jakaa osiin: esimerkiksi tyyppimääritykset annetaan usein omana dokumenttinaan. Tästä seuraa myös, että WSDL-kuvaus ei välttämättä ole yksi XML-dokumentti.

Erottelu palvelun rajapinnan määritykseen ja palvelun toteutuksen määritykseen on itse asiassa oleellinen WSDL-kuvauksen rekisteröinnin (UDDI) kannalta. Tästä lisää myöhemmin.

Simple Object Access Protocol (SOAP)

SOAP

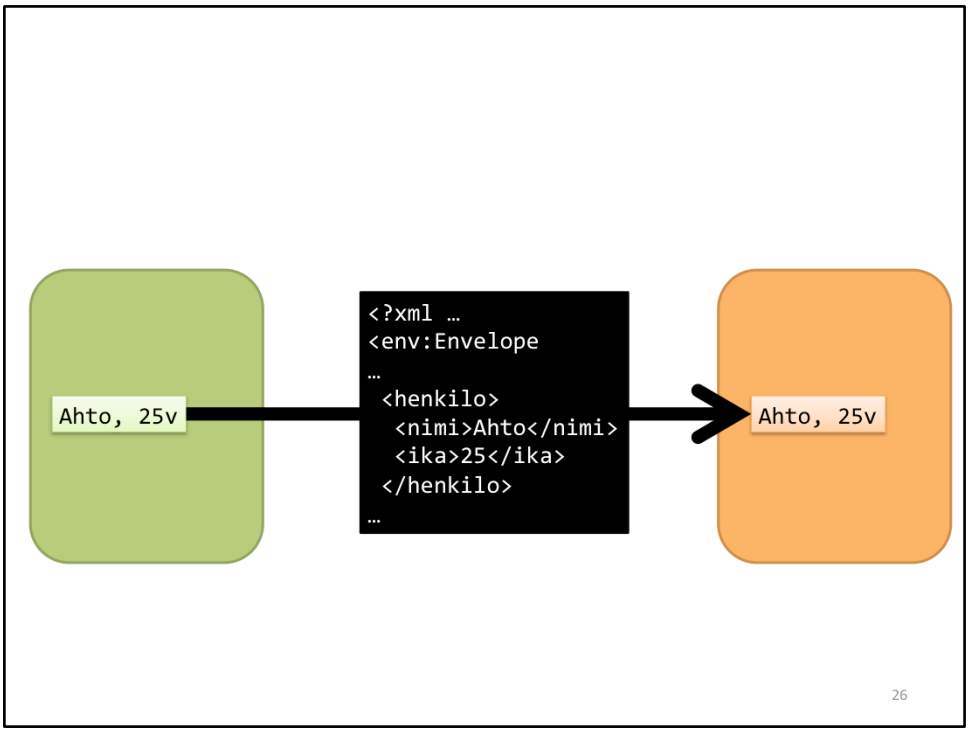
- SOAP is a protocol for cross-platform method invocations in XML
- The ”object” in the acronym is a bit misleading
  - object references are missing from SOAP
- Current W3C recommendation is SOAP 1.2
  - 1st edition published 2003, 2nd edition published 2007
  - Older version 1.1. is also still used
- Extensibility
  - features can be added as layered extensions
  - e.g., proposals for extending SOAP to carry digital signature information have been made

25

SOAPin uusin versio 1.2 on vuodelta 2003. Vaikka tämä versio onkin jo yleisesti käytössä ja myös W3C:n suositus, käytetään versiota 1.1 myös jonkin verran edelleen. SOAPia voidaan käyttää esim. tyypilliseen (yksisuuntaiseen) pyyntö-vaste –kommunikointiin, jolloin pyyntöä varten otetun HTTP-yhteyden aikana palautetaan myös vaste. Sitä voidaan käyttää myös kaksisuuntaiseen pyyntö-vaste –kommunikointiin (esim. HTTP-palvelin molemmissa päissä). SOAPia voidaan käyttää etäkutsujen merkkaamiseen (vrt. XML-RPC), mutta sen käyttö ei suinkaan rajoitu siihen. Verrattuna binäärisiin formaatteihin SOAP on melko tehoton. Tämä johtuu suurelta osin SOAPin XML-pohjaisuudesta: XML:n käsittely (jäsentäminen, generointi) on melko hidasta.

SOAP on riippumaton kuljetusprotokollasta. Yleisimmin käytössä on HTTP, mutta esimerkiksi SMTP tai FTP käyvät myös. Esimerkiksi asynkroninen viestinvälitys (esim. käyttämällä sopivia viestinvälitysprotokollia kuten SMTP) on toteutettavissa SOAPin avulla. SOAPin käyttömahdollisuudet eivät rajoitu pyyntö-vaste –kommunikointiin. Se voi toimia yhteisenä kutsuformaattina, jonka avulla voidaan esim. integroida heterogeenisia komponenttiteknologioita (CORBA, DCOM etc.) käyttäviä järjestelmiä. Se soveltuu myös hyvin Internetiä hyödyntäviin (löyhästi kytkettyihin) B2B- ja B2C-sovelluksiin. Lisäksi SOAPin arvo kevyenä protokollana on erityisen tärkeä pienille laitteille, joissa saattaa olla vain yksinkertainen HTTP-ympäristö ja XML-jäsennin, sillä niihin ei voi sisällyttää laajoja runtime-osia.

SOAPin yksi tärkeistä eduista on sen laajennettavuus. SOAPin ns. *header*-osa mahdollistaa sen. Sen avulla voidaan esimerkiksi viestiin liittää sovellusriippumatonta informaatiota.


Ahto, 25v
<?xml …
<env:Envelope
…
<henkilo>
<nimi>Ahto</nimi>
<ika>25</ika>
</henkilo>
…
Ahto, 25v
26

SOAP

- Fundamentally a stateless, one-way message exchange paradigm
- However, applications can create more complex interaction patterns
- Supports multiple transport protocols
  - HTTP is most commonly used

SOAP concepts

- An *envelope* defines a framework for a SOAP message
  - both application-specific and application-independent information can be included in the envelope
  - contains a required Body and an optional Header
- *Header* (optional) includes application-independent information
  - extension mechanism
  - used e.g. for passing directives or contextual information related to the processing of the message
- *Body* (mandatory) includes application-specific information
  - it contains the main end-to-end information
    - e.g., request/invocation and the parameters needed

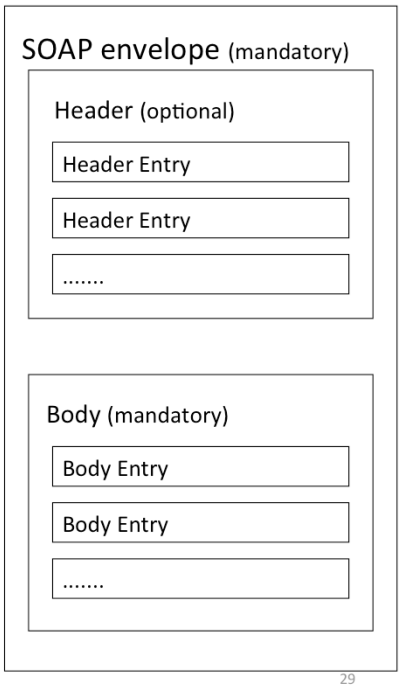
28

SOAPin pääosat ovat kirjekuori (envelope), otsikko (header) ja runko (body). Kirjekuori määrittelee kehyksen SOAP-viestille. Se sisältää otsikko- ja runko-osat. Otsikko ei SOAP-viesteissä ole pakollinen, mutta mikäli sitä käytetään, on sen oltava alussa ennen runko-osaa.
SOAPin otsikko-osa (header) mahdollistaa laajennettavuuden. Sen avulla voidaan esimerkiksi viestiin liittää sovellusriippumatonta informaatiota. Tällaista informaatiota on esimerkiksi turvalliseen viestinvälitykseen liittyvät asiat ja erilainen ohjausinformaatio viestin käsittelemiseksi.

SOAP kirjekuoren pakollisessa runko-osassa välitetään varsinainen sovellusspesifi tieto. Esimerkiksi pyyntö-vastaus –kommunikoinnin tapauksessa siinä välitetään varsinainen pyyntö (esim. operaatiokutsu) ja vastaus. Myös virheilmoitukset merkataan runko-osaan.

SOAP message structure:

```
<env:Envelope
    xmlns:env="http://www.w3.org/
    2003/05/soap-envelope">
  <env:Header>
    <!-- optional header blocks go here -->
  </env:Header>
  <env:Body>
    <!-- requests/responses -->
    <!-- or Fault elements go here -->
  <env:Body>
<env:Envelope>
```



29

SOAP-viestin juurielementti on *Envelope*. Sen alielementteinä ovat mahdollinen *Header* ja pakollinen *Body*. SOAP-viestin rakenne on varsin yksinkertainen. Viestin monimutkaisuus muodostuu *Header* ja *Body* osien sisäisestä rakenteesta.

SOAP spesifikaatio sisältää paljon optionaalisia sääntöjä. Tämä voi aiheuttaa ongelmia käytännössä: kaksi SOAP toteutusta eivät välttämättä toteuta kaikkia (tai samoja) optionaalisia piirteitä, mikä puolestaan voi aiheuttaa kommunikointiongelmia.

SOAP envelope

- A namespace identifying the SOAP version used is given in Envelope element
  - SOAP 1.2: http://www.w3.org/2003/05/soap-envelope
  - SOAP 1.1: http://schemas.xmlsoap.org/soap/envelope
- e.g.
  <env:Envelope
  xmlns:env="http://www.w3.org/2003/05/
  soap-envelope">

30

Envelope-elementillä on nimiavuuruusmääre, joka spesifioi käytettävän SOAP version. Esimerkiksi

<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">

kertoo, että kyseessä on SOAP 1.2 versio. Lisäksi siinä voidaan antaa muita nimiavaruusmäärityksiä. Käytettävä prefiksi (tässä *env*) voidaan luonnollisesti valita vapaasti. SOAPin eri versioiden käyttö saattaa aiheuttaa ongelmia. Esimerkiksi SOAP 1.2:a tukeva prosessori generoi virheen (VersionMismatch) kun sille annetaan SOAP 1.1 kirjekuori sille ominaisine nimiavaruusmäärityksineen.

```
<?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>Åke Jógvan Øyvind</n:name>
 </n:passenger>
 </env:Header>
 <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:return>
    <p:arriving>
      <p:airportChoices>
        JFK LGA EWR
      </p:airportChoices>
    </p:arriving>
  </p:return>
 </p:itineraryClarification>
 </env:Body>
</env:Envelope>
```

encodingStyle attribute

- `encodingStyle` attribute is used to define encoding rules used to serialize parts of a SOAP message
  - i.e. how to encode application-specific data in XML
- SOAP 1.2 Part 2 section 3 defines example SOAP encoding rules
  - Usage of this encoding scheme can be marked as env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
  - using this encoding scheme is optional, also other encoding schemes can be used
- may appear in specific parts of the header or the body in SOAP 1.2 or anywhere (also in envelope element) in SOAP 1.1

32

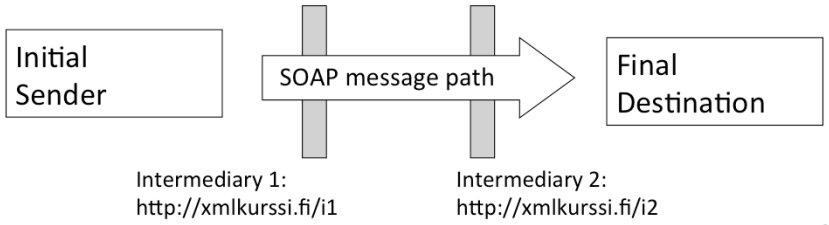
RPC:n toteuttamiseksi palvelun käyttäjän (Requestor) ja palvelun tarjoajan (Provider) tulee sopia mekanismista, jolla metodikutsu (erit. ohjelmien määrittelemät ja käyttämät tietotyypit) koodataan XML:ksi ja toisaalta miten se dekoodataan. Attribuuttia *encodingStyle* voidaan käyttää tämän sopimuksen määrittelemiseen. Toisin sanoen attribuuttia *encodingStyle* käytetään määrittelemään sarjallistamissääntöjen koodaus.

Esimerkki (SOAP 1.2 Part 0: Primer)

Tässä esimerkissä *encodingStyle* attribuutin arvo *http://www.w3.org/2003/05/soap-encoding* kertoo, että *changeReservation* rakenteen sarjallistamisessa on käytetty SOAPin määrittelemiä (SOAP Part 2 section 3) koodaussääntöjä. Vaikka SOAP spesifioikin nämä säännöt, niiden käyttö on optionaalista ja sovellukset voivat käyttää muitakin koodaussääntöjä sovellusspesifin datan koodaamiseksi SOAP-viestiin. *encodingStyle*-attribuuttia voidaan käyttää SOAP 1.2 versiossa sekä otsikko- että runko-osissa, mutta SOAP 1.1 sallii sen käytön kaikkialla myös *Envelope*-elementissä.

SOAP intermediaries

- A SOAP message can travel through multiple intermediaries before arriving to the final destination (message path)
- Useful for placing additional processing components on intermediaries without modifying the applications on the sender and ultimate destination ends
- A SOAP node can be the initial SOAP sender, an ultimate SOAP receiver, or a SOAP intermediary



34

SOAP-viesti voi kulkea viestin lähettäjältä vastaanottajalle useamman välittäjän kautta. Välittäjien käyttö onkin kätevä tapa välittää sama viesti useammalle taholle. Se myös antaa välittäjille mahdollisuuden käsitellä viestiä annettujen sääntöjen mukaisesti. Välittäjien käyttö Web-palvelukonseptissa on hyödyllistä: välittäjät voivat tarjota lisätoiminnallisuutta ja lisäarvopalveluita. Välittäjät yleisemminkin mahdollistavat lisäprosessoinnin SOAP-pohjaisessa viestivälityksessä edellyttämättä viestin lähettäjän ja vastaanottajan modifiointeja.

SOAP-viestin käsittelijöitä, joita siis voivat olla viestin lähettäjä, välittäjät ja vastaanottaja, kutsutaan SOAP-solmuksi (SOAP node).

SOAP Intermediaries (cont.)

- Message path
  - SOAP does not specify how a message path is determined and followed
  - message senders may or may not be aware of intermediaries in the message path
- Forwarding intermediaries
  - process the message according to the SOAP processing model
  - SOAP header blocks targeted at the SOAP intermediary MUST be removed from the SOAP message prior to forwarding
- Active intermediaries
  - process the message according to the SOAP processing model
  - may undertake processing not described by header blocks in the incoming message
  - the potential set of services provided by an active intermediary includes, but is not limited to: logging, security, content modification and tracing

35

Viestipolku kulkee viestin lähettäjältä välittäjien kautta viestin vastaanottajille. SOAP spesifikaatio ei ota kantaa siihen miten viestipolku määritellään ja miten viestit välitetään eteenpäin. Spesifikaatio kuitenkin määrittelee miten välittäjänä toimivan SOAP-solmun tulee käyttäytyä (käsitellä viesti) vastaanottaessaan SOAP-viestin. Viestin lähettäjä voi olla tietoinen viestipolun välittäjistä, mutta näin välttämättä aina ole.

Välittäjiä on kahdenlaisia: passiivisia (forwarding intermediaries) ja aktiivisia (active intermediaries). Passiiviset välittäjät prosessoivat viestin SOAPin sääntöjen mukaisesti (tärkeimmät asiat esitellään seuraavaksi) ja poistaa viestistä – tai tarkemmin sanottuna otsikosta - jo käsittelemänsä osat ennen viestin välittämistä eteenpäin. Aktiivinen välittäjä puolestaan sääntöjen mukaisen prosessoinnin lisäksi saattaa tarjota lisäpalveluita, joita ei ole määritelty viestissä. Tällaisia lisäarvopalveluita voivat olla esimerkiksi turvallisuuspalvelut, sisällön muuttaminen ja jäljitys.

SOAP Header

- Intended for adding features and functionalities
  - security, process flow, routing, and other QoS attributes
- If occurs in the envelope, it must be the first immediate child of the envelope
- Header entries (elements) may have the following attributes:
  - **encodingStyle**
  - **role** : defines who must process the header entry (intermediary destination)
  - **mustUnderstand**: defines whether the header entry is required or optional for the recipient (defined by role attribute) to be processed
    - required: mustUnderstand="true"
  - **relay**: indicates whether a SOAP header entry targeted at a SOAP receiver needs to be passed forward if not processed

36

Otsikko tarjoaa käytännössä SOAPin laajennosmekanismin. Otsikko-osassa voidaan antaa sovellusriippumatonta tietoa liittyen turvalliseen viestin välitykseen, viestin käsittelyn ohjaamiseen tai vaikkapa erilaisiin laatuattribuutteihin. Mikäli otsikko-osa annetaan, tulee sen olla ensimmäinen *Envelope*-elementin alielementti.

Otsikon osissa eli *Header*-elementin alielementeissä voidaan käyttää seuraavia attribuutteja:
-*encodingStyle* (käsitelty edellä)
-*role*: määrittelee SOAP-solmut, joiden tulee prosessoida viesti
-*mustUnderstand*: määrittelee tuleeko viestin käsittelevän SOAP-solmun prosessoida kyseinen otsikon osa. Mikäli prosessointia edellytetään, annetaan attribuutin arvoksi *true*. Attribuutti *mustUnderstand* suo käytännössä sovelluksille, jotka käyttävät vanhempaa spesifikaatiota, mahdollisuuden ”epäonnistua tyylikkäästi” kun saavat viestin, jota ne eivät ymmärrä.
-*relay*: määrittelee tuleeko SOAP-viestin vastaanottajan (SOAP-solmu) välittää ko. otsikon osan tieto eteenpäin mikäli se ei ole käsitellyt tätä otsikon osaa. Tiedon eteenpäin välittäminen tarkoittaa, että vastaava otsikon osa generoidaan eteenpäin välitettävään viestiin. Myös *relay*-attribuutti saa boolean-arvot *true* tai *false*. Mikäli *relay*-attribuutti puuttuu, on käyttäytyminen sama kuin jos sen arvoksi olisi annettu *false*.

```
<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>Åke Jógvan Øyvind</n:name>
 </n:passenger>
</env:Header>
```

37

SOAP roles

- http://www.w3.org/2003/05/soap-envelope/role/**next**
  - targets a header entry at whichever processor receives it
  - all SOAP processing nodes must support the special role "next", which makes it possible to target a header block at the next processor in a message path.
  - each SOAP node MUST act on this role
- http://www.w3.org/2003/05/soap-envelope/role/**none**
  - SOAP nodes MUST NOT act on this role
- http://www.w3.org/2003/05/soap-envelope/role/**ultimateReceiver**
  - the ultimate receiver MUST act on this role
  - indicates that a header block is targeted at the ultimate SOAP receiver in a
  - message path

38

SOAPin otsikon elementeissä käytettävä *role*-attribuutti, joka määrittelee minkä SOAP-solmujen tulee prosessoida ko. otsinkon osa, voi saada kolmenlaisia arvoja:
-*next*: vastaanottavan viestin käsittelijän (SOAP-solmu) viestipolulla tulee prosessoida ko. otsikon osa
-*none*: minkään viestin vastaanottavan SOAP-solmun ei tule käsitellä ko. otsikon osaa
-*ultimateReceiver*: ainoastaan viestin varsinaisen vastaanottajan tulee käsitellä ko. otsikon osa. Tämä on myös oletusarvo/ oletuskäyttäytyminen mikäli attribuuttia *role* ei ole annettu.

SOAP body

- body element is the actual payload of a SOAP message
- Contains application-specific data
  - e.g. requests and responses
- Intended for the ultimate receiver

39

SOAP-viestin runko-osa sisältää sovellusspesifistä dataa. Siihen merkataan esimerkiksi pyyntö-vaste –kommunikoinnin pyyntöosat ja vastaavasti vastausosat. Samoin virheilmoituksen välitetään runko-osassa. Runko-osa on aina pakollinen SOAP-viestissä.

Esimerkki onnistuneesta pyyntö-vaste -kommunikoinnista:

SOAP pyyntö (request):

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
   <env:Body>
                  <m:GetPrice
                        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding”
                        xmlns:m="http://example.org/2001/06/quotes">
                     <symbol>DIS</symbol>
                  </m:GetPrice>
   </env:Body>
</env:Envelope>
```

SOAP vastaus (response):

```
<env:Envelope xmlns:env=" http://www.w3.org/2003/05/soap-envelope">
<env:Body>
                  <m:GetPriceResponse env:encodingStyle=”http://www.w3.org/2001/09/soap-encoding”
                                    xmlns:m="http://example.org/2001/06/quotes">
        <Price>34.5</Price>
                  </m:GetPriceResponse>
</env:Body>
</env:Envelope>
```

- SOAP request:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Body>
  <m:GetPrice
     env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
     xmlns:m="http://example.org/2001/06/quotes">
   <symbol>DIS</symbol>
  </m:GetPrice>
 </env:Body>
</env:Envelope>
```

- SOAP response:

```
<env:Envelope xmlns:env=" http://www.w3.org/2003/05/soap-envelope">
 <env:Body>
  <m:GetPriceResponse
     env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
     xmlns:m="http://example.org/2001/06/quotes">
   <Price>34.5</Price>
  </m:GetPriceResponse>
 </env:Body>
</env:Envelope>
```

Esimerkki onnistuneesta pyyntö-vaste -kommunikoinnista

SOAP Fault

- Fault element is used to carry error information
- Contents
  - Mandatory **Code** element: A local name of the code, includes fault codes
    - subelements Value ja Subcode
  - Mandatory **Reason** element: A human readable explanation of the fault
  - Optional **Node** element: Intended to provide information about which SOAP node on the SOAP message path caused the fault to happen
  - Optional **Role** element: Identifies the role the node was operating in at the point the fault occurred
  - Optional **Detail** element: Intended for carrying application specific error information related to the SOAP Body

Virheet voidaan merkata SOAP-viestiin spesifikaation määrittelemällä tavalla käyttäen *Fault*-elementtiä. *Fault*-elementti sisältää (tai voi sisältää) seuraavat alielementit:
-*Code*: Tämä pakollinen elementti sisältää virhekoodin. Se sisältää puolestaan pakollisen *Value*-elementin ja optionaalisen *Subcode*-elementin. SOAP spesifikaatio määrittelee sisällön elementille *Value*. Spesifikaatio määrittelee pienen joukon virhekoodeja korkean tason SOAP-virheille. Nämä virhekoodit esitellään seuraavaksi.
-*Reason*: Tämä pakollinen elementti sisältää virheen kuvauksen.
-*Node*: Tämän optionaalisen elementin avulla voidaan identifioida se SOAP-solmu, joka aiheutti virheen. Tämä solmu yksilöidään URI:n avulla
-*Role*: Tämä optionaalinen elementti identifioi viestiä operoivan SOAP-solmun roolin virheen tapahtuessa.
-*Detail*: Tätä optionaalista elementtiä voidaan käyttää kuljettamaan sovellusspesifistä virhedataa. Tällä elementillä voi olla attribuutteja ja alielementtejä.

Fault codes (*Value* subelement of *Code* element):

Fault code	Description
VersionMismatch	An invalid namespace found for the SOAP envelope element
DataEncodingUnknown	A header/body block targeted at the current SOAP node is scoped with a data encoding that the current node does not support
MustUnderstand	An element with mustUnderstand=”1” was not understood
Sender	The message was incorrectly formed or contained incorrect information
Receiver	The message could not be processed, e.g., a problem with the server

42

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env=”http://www.w3.org/2003/05/soap-envelope” xmlns:xml="http://
www.w3.org/XML/1998/namespace">
 <env:Header>
  <env:Upgrade>
   <env:SupportedEnvelope qname="ns1:Envelope“
     xmlns:ns1="http://www.w3.org/2003/05/soap-envelope"/>
   <env:SupportedEnvelope qname="ns2:Envelope"
     xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope/"/>
  </env:Upgrade>
 </env:Header>
 <env:Body>
  <env:Fault>
   <env:Code>
    <env:Value>env:VersionMismatch</env:Value>
   </env:Code>
   <env:Reason>
    <env:Text xml:lang="en">Version Mismatch</env:Text>
   </env:Reason>
  </env:Fault>
 </env:Body>
</env:Envelope>
```

43

Esimerkki (W3C, SOAP 1.1, Part 1): virhekoodi (Code-elementti) kertoo, että kyse on versio-ongelmasta. Otsikon Upgrade-lohko indikoi, että SOAP-solmu tukee sekä SOAP versiota 1.2 että versiota 1.1 mutta preferoi versiota 1.2.

SOAP Message Exchange Patterns (MEP)

- SOAP 1.2 Part 2 describes two Message Exchange Patterns
- **Request-Response**
  - Echange two messages between nodes along one SOAP message path, one in each direction
- **Response**
  - Non-SOAP request – SOAP response

SOAP MEPs

- SOAP 1.2 Part 2 provides quidance when to use request-respond and when respond MEPs
- **Request-Response**, use when
  - information resource is manipulated
- **Response**, use when
  - an application is assured that the message exchange is for the purposes of information retrieval (safe!)
  - information resource is "untouched" as a result of the interaction

45

SOAP 1.2 spesifikaation osa 2 (Part 0) antaa ohjeistusta sille, milloin eri viestinvälitysmalleja (MEPs) kannattaa käyttää. Ne ovat luonnollisesti vain suosituksia, vaikkakin suositus on annettu melko vahvaan sävyyn.

SOAP Respond viestinvälitysmallia tulisi käyttää silloin, kun kommunikoinnin kohdetta ei ole tarkoitus muuttaa interaktion toimesta. Se on tiedon hakemista varten. HTTP spesifikaatio kuvaa tällaisia yhteyksiä turvallisiksi.

SOAP Request-Respond viestinvälitysmallia tulee käyttää silloin, kun on tarvetta informaatiolähteen manipuloimiseen. Kannattaa huomata, että HTTP POST sidonta käy siis kaikissa tapauksissa.

SOAP HTTP binding

- SOAP messages can be transported on top of any transport protocol
  - However, most often HTTP is used
- HTTP request-respond MEP: the client identifies the server via a URI, connects to it using the underlying TCP/IP network, issues a HTTP request message and receives a HTTP response message over the same TCP connection
- HTTP bindings with SOAP
  - allows the exchange of SOAP messages either as payload of a HTTP POST request and response, or as a SOAP message in the response to a HTTP GET.
    - HTTP binding with the SOAP Respond MEP (i.e., non-SOAP message acting as a request followed by a SOAP message acting as a response) is restricted to the HTTP GET method
    - HTTP binding with the SOAP Request-Response MEP is restricted to the HTTP POST method

46

HTTP kommunikoi käyttäen TCP/IP-protokollaa. Asiakas identifioi palvelimen URI:n avulla, ottaa siihen yhteyden sekä lähettää HTTP kutsun
SOAP voidaan sitoa useampaan protokollaan, mutta yleisin käytännössä on HTTP. SOAP-spesifikaatio määrittelee HTTP-sidonnan. Siinä SOAP pyyntö-vaste kommunikointi perustuu HTTP POST metodin käyttöön, kun taas yksinkertainen SOAP vaste (pyyntö tapahtuu muuten kuin SOAPia käyttäen) on sidottu HTTP GET metodin käyttöön.

- HTTP Request

```
POST /item HTTP/1.1
Host: 189.123.345.239
Content-Type: text/plain
Content-Length: 200
```

- HTTP Response

```
200 OK
Content-Type: text/plain
Content-Length: 200
```

HTTP kommunikoi käyttäen TCP/IP-protokollaa. Asiakas identifioi palvelimen URI:n avulla, ottaa siihen yhteyden sekä lähettää HTTP kutsun. Esimerkki: W3Schools, http://www.w3schools.com/soap/soap_httpbinding.asp

Using HTTP GET and HTTP POST for SOAP MEPs

- Using HTTP GET
  - HTTP Accept header is used to indicate the preferred representation of the resource being requested, e.g. “application/soap+xml”
- Using HTTP POST
  - HTTP Content-type header must be "application/soap +xml“
    - charset parameter can take either value “utf-8” or “utf-16”
  - the combination of HTTP POST and Host headers represents URI of the resource

48

- **SOAP Response MEP**

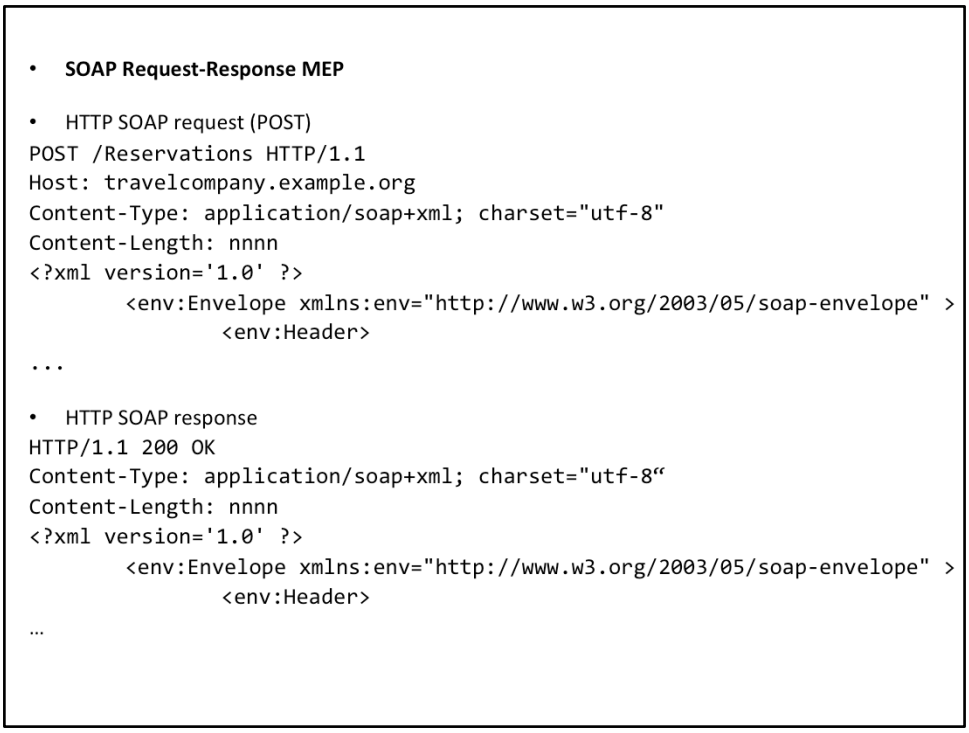
- HTTP SOAP request (GET)

```
GET /travelcompany.example.org/reservations?code=FT35ZBQ
HTTP/1.1 Host: travelcompany.example.org
Accept: text/html;q=0.5, application/soap+xml
...
```

- HTTP SOAP response

```
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/2003/05/soap-
envelope">
 <env:Header>
  <m:reservation
     xmlns:m="http://travelcompany.example.org/reservation"
...
```

SOAP 1.2, Part 0: Primer



Esimerkki 2. HTTP POST-pyyntö:

Tässä HTTP Content-Type kentän arvon tulee aina olla ”application/soap+xml”. Sen optionaalinen *charset*-parametri voi saada joko arvon ”utf-8” tai ”utf-16”. Itse kutsuttavan resurssin URI muodostuu POST ja Host kenttien arvoista. Näin ollen alla olevassa esimerkissä URI on travelcompany.example.org/Reservations

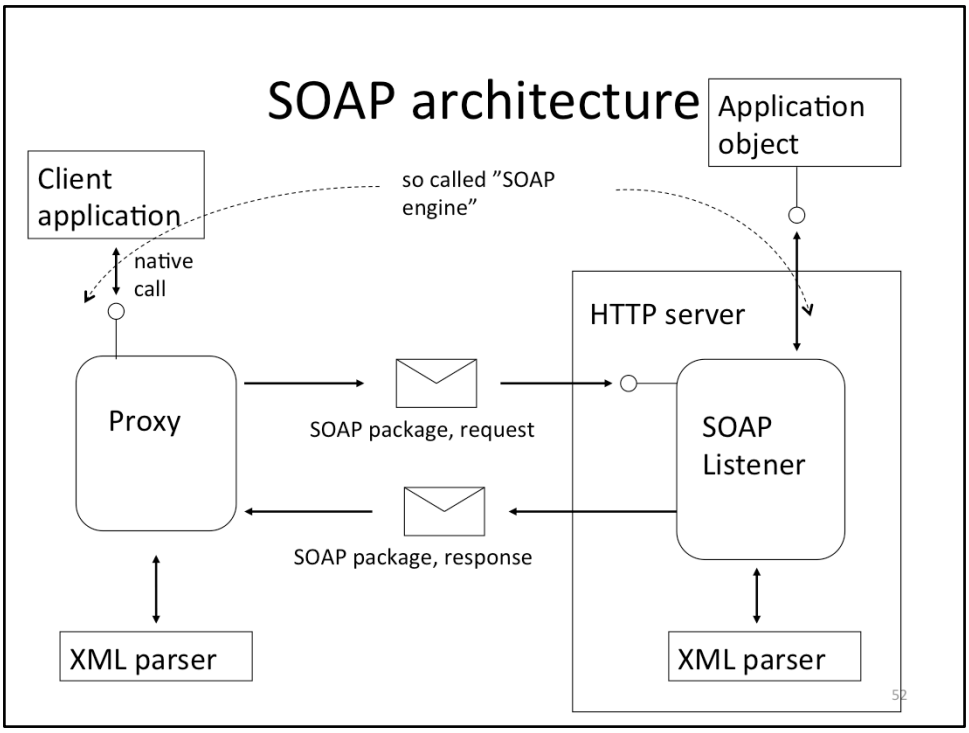
HTTP respond codes

- 2xx (successful)
- 3xx (redirection)
- 4xx (client error)
- 5xx (server error)
- Example status codes of errors
  - 401: the request requires HTTP authentication
  - 404: the requested resource is not available
  - 405: the peer HTTP server does not support the requested HTTP method at the given request URI
  - 500: an error inside the HTTP server which prevented it form fulfilling the request
  - 503: the HTTP server is temporarily overoaded, and unable to handle the request

51

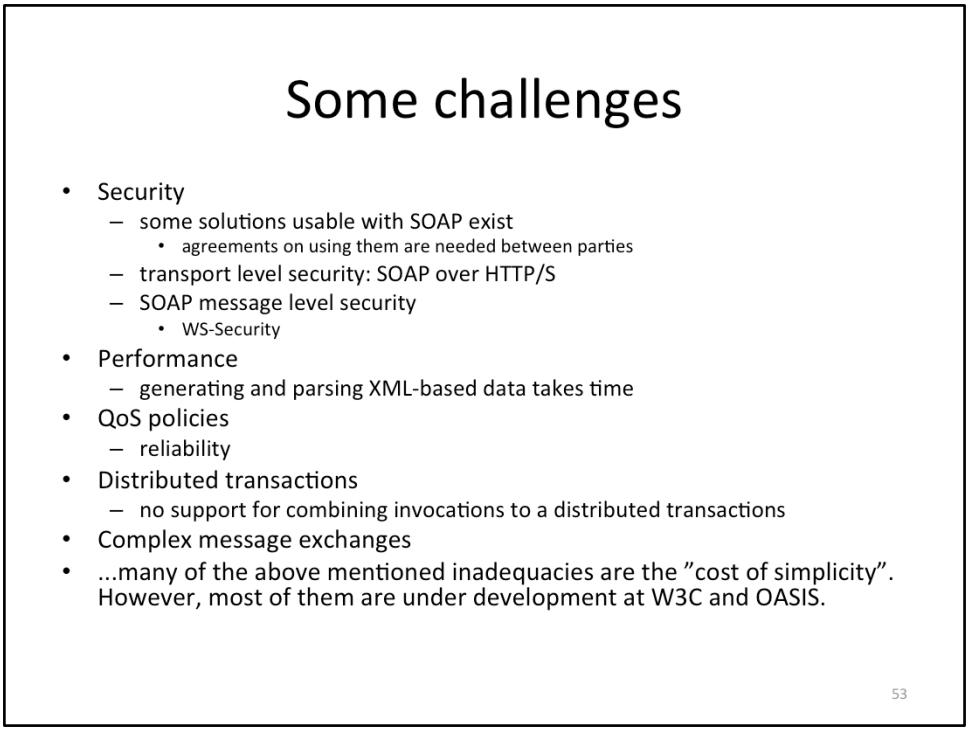
HTTP vastauskoodeista voidaan päätellä onnistuiko asiakkaan lähettämän viestin vastaanottaminen, ymmärtäminen ja onko se hyväksytty (2xx -alkuiset arvot) ja toisaalta mikä oli mahdollisen virheen syy. Esimerkiksi palvelinpään prosessointiongelmasta johtuva virhe voitaisiin ilmoittaa seuraavalla tavalla:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
                <env:Body>
                                <env:Fault>
                                                <env:Code>
....
```



SOAP Listener ottaa vastaan SOAP-paketteina saapuvia viestejä ja välittää ne edelleen itse palvelulle käännettyään ne ko. palvelun natiivikielelle (esim. Java). Asiakassovelluksen (client application) tarvitsee ainoastaan tietää palvelun osoite sekä palvelun ymmärtämän/ymmärtämien viestin/viestien muoto/muodot. Kun palvelu on prosessoinut viestin ja muodostanut mahdollisen vastauksen, se tulee niin ikään muuttaa SOAP viestiksi ja välittää asiakkaalle.

Web-palveluissa palvelun käyttäjä (Requestor) kutsuu (bind operaatio) valittua palvelua (Provider) lähettämällä SOAP-viestin. Yksinkertainen proxy jäsentää palvelun kuvauksen, joka annetaan WSDL-muodossa; WSDL kuvaus kertoo miten kyseistä palvelua kutsutaan. WSDL-kuvauksia onkin verrattu IDL-kuvauksiin: WSDL on Web-palveluille sitä mitä IDL:t ovat CORBAlle. Koska WSDL on ohjelmallisesti luettavaa ja se sisältää tarvittavat tiedot, voidaan siitä itse asiassa generoida asiakaspään (client) stub koodia, joka toimii proxynä. WSDL-kuvausta voidaan hyödyntää myös palvelinpäässä: siitä voidaan myös generoida koodia, jonka palvelinsovelluksen toteuttaja voi sopivasti laajentaa. WSDL-kieleen perehdymme seuraavaksi.



Kun kyse on kommunikoinnista Internetiä hyödyntäen, nousee joissain tapauksissa (erityisesti liiketoiminnan kannalta kriittisissä sovelluksissa) tietoturva oleelliseksi. Tähän liittyviä asioita ovat autentikointi, tiedon salaus, digitaaliset allekirjoitukset jne. SOAP itsessään ei tarjoa tukea näiden toteuttamiseksi. On kuitenkin esitetty ehdotuksia esimerkiksi siitä, miten digitaaliset allekirjoitukset voitaisiin liittää SOAP-viesteihin. SOAPin laajennettavuus (header) mahdollistaa tämän. Web-palveluissa tämä voidaan hoitaa eri kerroksessa (kts. Layers of Web Services) eri tavoin. SOAP-viesteissä voidaan digitaalisten allekirjoitusten lisäksi antaa esimerkiksi WS-Security specifikaation (http://www-106.ibm.com/developerworks/webservices/library/ws-secure/) mukaisia lisäyksiä autentikoinnin, koskemattomuuden ja luottamuksellisuuden varmistamiseksi. Lisäksi SOAPin pohjalta on kehitetty ebXML Message Service, jota käytetään ebXML-konseptissa. Tämän ja turvalliseen viestinvälitykseen palataan myöhemmin.

Web-palveluiden kannalta eri laatuattribuutit (vasteajat, hinta ja muut liiketoiminnan kannalta oleelliset vaatimukset), kuten luotettavuus, ovat myös tärkeitä. SOAP ei itsessään tue myöskään näiden merkkausta. Yksi oleellinen asia Web-palveluiden kannalta on myös transaktioiden hallinta: sekvenssi yksittäisten palvelujen suorittamista operaatiosta halutaan koostaa yhdeksi

SOAP data types

- Simple types defined by XML Schema
  - http://www.w3.org/TR/2000/WD-xmlschema-0-20000407/
  - string, boolean, float, double, decimal, timeInstant, timePeriod, month, year, century, recurringDate, recurringDay, timeDuration, recurringDuration, binary, uriReference, language, Name, QName, NCName, integer, nonPositiveInteger, negativeInteger, long, int, short, byte, nonNegativeInteger, unsignedLong, unsignedInt, unsignedShort, unsignedByte, positiveInteger, date, time

54

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>….</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">
        ...
 </env:Body>
</env:Envelope>
```

55